Ein echtzeitfähiges System zur Gewinnung von Tiefeninformation aus Stereobildpaaren für konfigurierbare Hardware

D I S S E R T A T I O N

zur Erlangung des akademischen Grades Dr. rer. nat.
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Dipl.-Inf. Maximilian Buder

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Elmar Kulke

Gutachter:
1. Prof. Dr.-Ing. Beate Meffert
2. Prof. Dr. habil. Herbert Jahn
3. Prof. Dr.-Ing. habil. Rainer G. Spallek

eingereicht am: 25.04.2013

Tag der mündlichen Prüfung: 30.04.2014

Für Hanna, Philipp, Markus und Sabine

Abstract

This work presents a realtime stereo image matching system that takes advantage of a global image matching method. The system is designed to provide depth information for mobile robotic applications. Typical tasks of the proposed system are to assist in obstacle avoidance, SLAM and path planning of mobile robots, that pose strong requirements on the size, energy consumption, reliability, frame rate and quality of the calculated depth map. Current available systems either rely on active sensors or on local stereo-image matching algorithms. The first are only suitable in controlled environments while the second suffer from low quality depth-maps. Top ranking quality results are only achieved by an iterative approach using global image matching and colour segmentation techniques which are computationally demanding and therefore difficult to be executed in real time. Attempts were made to still reach real-time performance with global methods by simplifying the routines but led to degraded depth maps which are at the end almost comparable with local methods. An equally named semi-global algorithm was proposed earlier, that shows both very good image matching results and relatively simple execution at the same time. A memory efficient variant of the Semi-Global Matching algorithm is presented and adopted for an implementation based on reconfigurable hardware that is suitable for real-time operations in the field of robotics. It will be shown that the modified version of the efficient Semi-Global matching method is delivering equivalent result compared to the original algorithm. The complete design has been implemented within a hardware development framework that is also reviewed.

Zusammenfassung

Diese Arbeit befasst sich mit der Entwicklung eines echtzeitfähigen Systems zur Erstellung von Tiefeninformation aus Stereobildpaaren, das in einer Reihe von Anwendungen zur dreidimensionalen Vermessung des Raumes herangezogen werden kann. Als Hauptanwendungsgebiete sind in erster Linie mobile Robotikapplikationen vorgesehen, die sehr strenge Anforderungen sowohl bezüglich des Ressourcenverbrauchs als auch im Hinblick auf die Messeigenschaften und das Laufzeitverhalten stellen. Ein Merkmal des in dieser Arbeit entworfenen Systems ist die in Echtzeit stattfindende Ausführung der verwendeten Algorithmen in Kombination mit sehr guten Messeigenschaften. Das verwendete Stereo-Matching-Verfahren basiert auf einem globalen Ansatz und liefert im Vergleich zu den alternativen echtzeitfähigen Methoden sehr gute Ergebnisse. Im Vordergrund steht dabei der Semi-Global-Matching-Algorithmus. Aufgrund der Komplexität globaler Ansätze finden in Echtzeitapplikationen nur lokale Stereo-Verfahren Verwendung. Lokale Verfahren liefern jedoch im Vergleich zu den globalen Methoden qualitativ schlechte Disparitätskarten. Ein neuer globaler Matching-Algorithmus Efficient-Semi-Global-Matching (eSGM) wird vorgestellt und in das Konzept für mobile Robotikanwendungen umgesetzt. Wegen der begrenzten Ressourcen der realen Hardware wurde eine Weiterentwicklung des eSGM-Algorithmus für die Realisierung genutzt. Abschließend wird das System anhand der drei Kerneigenschaften Laufzeit, Ressourcenverbrauch und Qualität der Tiefeninformation gegenüber den Verfahren nach dem Stand der Technik bewertet. Der in dieser Arbeit vorgestellte FPGA-Ansatz, die eingesetzte Entwurfsmethode und die vorgestellten Algorithmen ermöglichten es, ein leistungsfähiges Stereo-Bildverarbeitungssystem zu entwickeln, das den hohen Anforderungen bezüglich des Laufzeitverhaltens und der Qualität des Ergebnisses gerecht wird.

Inhaltsverzeichnis

1. Einleitung

Der Gegenstand der hier vorliegenden Arbeit ist die automatische Gewinnung von Informationen über die Position von Objekten im dreidimensionalen Raum aus Stereobildaufnahmen. Dies ist motiviert durch die allgemeinen Bedeutung, die die digitale Stereophotogrammetrie sowohl für die Fernerkundung als auch für Industrie- bzw. Robotikanwendungen im Nahbereich hat. Der gesellschaftliche Nutzen der Photogrammetrie ergibt sich insbesondere aus den Lösungen, die sie der Wissenschaft, dem Militär und der Wirtschaft anzubieten in der Lage ist. So ist es mit Hilfe der Luft- und Satellitenbildphotogrammetrie möglich, ein großes Gebiet relativ schnell geometrisch absolut zu erfassen. Eine der allgemein bekannten Aufgabenstellungen dafür ist die Erstellung topographischer Landkarten; modernere Anwendungsbereiche sind hier der Umwelt- und Katastrophenschutz, Klimaschutz, die Raumordnung und die Erstellung interaktiver 3D-Welten. Im industriellen Umfeld ist die berührungslose Vermessung von Objekten zur Qualitätssicherung eine typische Anwendung. Ein weiteres aktuelles Thema ist die Weiterentwicklung unbemannter und computergestützter Systeme, für die die räumliche Erfassung der Umgebung häufig eine zwingende Voraussetzung ist.

1.1. Motivation

Mit der Einführung der digitalen Bildgewinnung und -verarbeitung sind zahlreiche aufwändige Arbeitsschritte überflüssig geworden, die mit den analogen Aufnahme- und Verarbeitungsvorrichtungen einhergingen. In den analogen Anfängen mussten die sich überlappenden Aufnahmen zeitintensiv manuell registriert werden. Zuvor waren die Bilder ebenfalls manuell entwickelt und in physischer Form transportiert worden. Heutige Aufnahmesysteme hingegen erfassen die Bildinformation mit Hilfe digitaler Zeilen- bzw. Flächensensoren. Die Aufnahmen stehen sofort für die anschließende Verarbeitung zur Verfügung. Zudem können die Bilder beliebig oft und verlustfrei zwischengespeichert und vervielfältigt werden. Die Bildaufnahmen müssen zudem nicht zwingenderweise von einem Operator gehandhabt werden; sowohl die Aufnahme als auch die Verarbeitung der Bildinformation geschehen vollautomatisch und die gewünschten Ergebnisse können direkt an den Bedarfsträger übermittelt werden. Diese prinzipielle Vorgehensweise stellt den heutigen Stand der Technik dar. Der Forschungs- und Entwicklungsschwerpunkt der letzten Jahre hat sich im Wesentlichen auf die Verbesserung der digitalen Verarbeitung und der Vergrößerung der Auflösung konzentriert. Ziel der Bemühungen ist es unter anderem, die Verarbeitung zu beschleunigen und gleichzeitig das Endergebnis zu verbessern.

Der im Vordergrund stehende Anwendungsschwerpunkt dieser Arbeit ist die Rekonstruktion von Tiefeninformation im Nahbereich, wobei auf die Ursprünge aus der Fernerkundung im weiteren Verlauf dieser Arbeit Bezug genommen wird. Beide Szenarien unterscheiden sich in den Voraussetzungen und den Anforderungen. In der Fernerkundung werden für die Erstellung qualitativ hochwertiger Höhen- und Oberflächenmodelle in der Regel optisch und mechanisch sehr aufwendige und kostenintensive Luftbildkameras eingesetzt, die allein aufgrund ihrer Größe nicht überall eingesetzt werden können. Heutige Industriekamerasysteme sind hingegen relativ kostengünstig und kompakt, so dass deren Integration in eine Vielzahl von Anwendungen, die den Nahbereich umfassen, interessant ist.

Dazu zählen unbemannte und autonom agierende Flugdrohnen (UAV, *unmanned arial vehicles*), die zum Beispiel zur Exploration in unbekanntem, menschenfeindlichen Terrain eingesetzt werden können. Für den Fall, dass optische Sensoren zur Verfügung stehen, können diese verwendet werden, um zum Beispiel Objekte im Raum als Hindernisse zu identifizieren oder Zielobjekte zu verfolgen. Weiterhin können die optischen Sensoren ergänzend zur Bestimmung der Lage des UAVs genutzt werden, um bei fehlender oder unzureichender Information durch die GNSS-Sensoren (Global Navigation Satellite System) das Fluggerät weiterhin sicher zu manövrieren. Letzteres ist insbesondere in Innenräumen ein wichtiges Kriterium. Eine weitere Voraussetzung dafür, dass ein autonomes System sich sowohl im Freien als auch in Gebäuden sicher bewegen kann, ist die Verarbeitung der Sensordaten in Echtzeit auf dem Gerät; im Fall von Fluggeräten bedeutet dies, dass die Algorithmen den Flugbetrieb nicht negativ beeinflussen dürfen.

Das Deutsche Zentrum für Luft- und Raumfahrt e.V. (DLR) ist an mehreren Projekten beteiligt, die sich zum einen mit der Entwicklung hochauflösender Luft- und Satellitenbildkameras und zum anderen mit der Erstellung daraus abgeleiteter Produkte beschäftigen. Ein Schwerpunkt sind in diesem Zusammenhang Forschungs- und Entwicklungsarbeiten, die sich mit der Etablierung effizienter und robuster Algorithmen befassen. Das DLR-Projekt Echtzeit-3D ist solch ein Vorhaben, dessen Aufgabe darin besteht, geeignete Verfahren zur Mehrbildauswertung zu finden, zu analysieren und weiterzuentwickeln. Im Vordergrund steht dabei immer deren effiziente Ausführung, ohne die Qualität der gewonnenen Information zu vernachlässigen.

Im Fall der hochauflösenden Luft- und Satellitenbildauswertung kann durch den Einsatz spezieller Hardware die Verarbeitung der großen Datenmengen beschleunigt werden. Die relativ hohen Datenmengen sind eine große Herausforderung an die Bandbreite der Schnittstellen und die Kapazität der Speichereinheiten. Im Vergleich dazu sind die Bildauflösungen in Robotik- bzw. Automatisierungsanwendungen eher moderat. Eine übergeordnete Rolle spielt das Echtzeitverhalten der Anwendung, so dass die Datenraten der Verarbeitungs- und Speichereinheiten in den Vordergrund treten. Zusätzlich stellen mobile Anwendungen wie zum Beispiel die UAVs erhöhte Anforderungen an den Energie- und Platzbedarf der Verarbeitungshardware.

Die klassische Hardwareplattform für die Bildverarbeitung ist in vielen Bereichen nach

wie vor der Mehrzweckprozessor, der den Bildverarbeitungsalgorithmus typischerweise sequentiell abarbeitet. Ausschlaggebend für die Wahl eines solchen Prozessors sind die geringen Investitionskosten und das einfache Programmiermodell. Für die Verarbeitung großer Bildverbände aus der Fernerkundung benötigen heutige Prozessoren allerdings Tage bis Wochen. In der Robotik gilt Ähnliches, demzufolge ist mit Standardprozessoren die Forderung nach Echtzeitfähigkeit häufig nicht realisierbar. Durch Parallelisierung der Bildverarbeitung kann die Berechnungsdauer erheblich verkürzt bzw. Echtzeitfähigkeit hergestellt werden. Es ist daher zweckmäßig, die Teile des Algorithmus, die sich nebenläufig ausführen lassen, von Hardware berechnen zu lassen, die die Parallelverarbeitung unterstützt. Integrierte Schaltkreise, die vollständig nach Kundenwunsch gefertigt wurden (ASICs, Application Specific Integrated Circuit), verfügen über das größte Potential zur Beschleunigung der Anwendung und sind aufgrund ihres relativ geringen Energie- und Platzverbrauchs zudem sehr gut für Robotikanwendungen geeignet; insbesondere in Forschungs- und Entwicklungsprojekten verbietet sich jedoch der ASIC aufgrund der sehr hohen Entwicklungskosten. Praktikable Alternativen sowohl zum Vollkunden-IC wie zum Standardprozessor sind eine Vielzahl von Prozessorerweiterungen, rekonfigurierbare Hardware und Graphikprozessoren. Aufgrund ihrer sich stetig weiter entwickelnden Leistungsfähigkeit und der Verschiedenheit ihrer Eigenschaften werden diese alternativen Lösungen den gestellten Anforderungen auch unterschiedlich gerecht.

1.2. Ziel der Arbeit

Diese Arbeit erhebt den Anspruch, ein neuartiges System zur Gewinnung von Information über den dreidimensionalen Raum aus Stereobildaufnahmen vorzustellen. Das System ist in der Lage, hochwertige Tiefenbilder zu erstellen, und hat gleichzeitig die Eigenschaft, sehr effizient in der Ausführung zu sein. Bislang sind Methoden zur Korrespondenzanalyse aufgrund ihrer algorithmischen Komplexität erst nach einigen Sekunden in der Lage, ein qualitativ hochwertiges Tiefenbild zu liefern. Andererseits erzeugen Systeme, die die Forderung nach Echtzeit erfüllen, Tiefenkarten, die nach heutigem Verständnis von minderer Qualität sind. Insbesondere für Robotikapplikationen sind die damit erstellten Tiefenbilder nicht ausreichend. Auf der anderen Seite stehen die innovativen und qualitativ hochwertigen Stereo-Matching-Verfahren zur Verfügung, die aufgrund ihrer Komplexität noch keinen Eingang in echtzeitfähige Systeme zur Gewinnung von Tiefeninformation aus Stereobildaufnahmen – im Folgenden auch als Stereo-Vision-Systeme bezeichnet – gefunden haben.

Der wissenschaftliche Beitrag dieser Arbeit liegt im Entwurf und der Realisierung eines neuen Verfahrens zur Bestimmung der Tiefeninformation aus Stereobildaufnahmen. Der neu entwickelte Algorithmus liefert die Grundlage dafür, sowohl eine hochwertige Tiefenkarte zu erstellen als auch diese in Echtzeit berechnen zu können.

Das vorgestellte Stereo-Vision-System kann in einer Vielzahl von Anwendungen eingesetzt werden: Zum einen als unterstützende Subkomponente in einer komplexen Verarbeitungskette, wie sie mobile oder stationäre Robotikanwendungen aufweisen,

und zum anderen als Hauptbestandteil in Prozessierungsketten für Bildverarbeitung.

1.3. Aufbau der Arbeit

Untersucht werden alle für das Gesamtsystem relevanten Bereiche. Sowohl der theoretische Teil als auch die Umsetzung und die Anwendung folgen dem Aufbau des Systems und beginnen mit der Bildentstehung im Sensor. Der Schwerpunkt liegt auf den Verfahren zur Stereobildauswertung. Die in dieser Arbeit vorgestellten Verfahren gehören demnach zur Gruppe der berührungslosen Messmethoden, die auf passiver Triangulation basieren (Abb. 1.1). Andere Verfahren, insbesondere solche, die in monokularen Aufbauten Tiefeninformationen anhand von Schattierungen, Textur, Fokus, Überdeckung und Details etc. extrahieren, sollen hier nicht betrachtetet werden. In der neueren Literatur bieten R. Szeliski [103] und A. Saxena et al. [96] eine Einführung zu diesen Methoden.

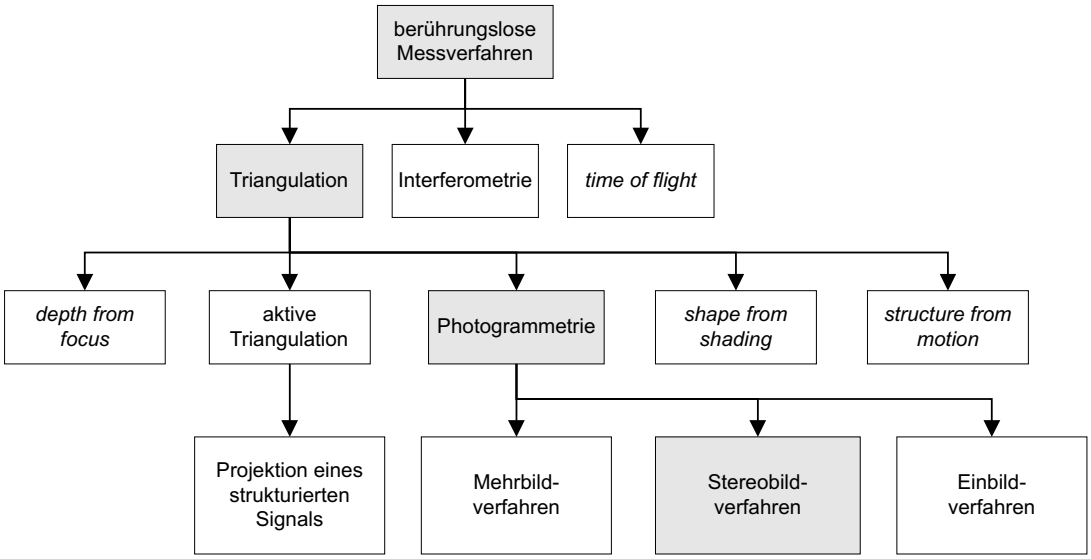


Abbildung 1.1.: Eine Übersicht berührungsloser Messverfahren für die Bestimmung der räumlichen Tiefe nach B. Jahne et al. [59]

Zur Einleitung werden daher in den nachfolgenden Kapiteln 2, 3 und chapter 4 sowohl die Grundlagen aus algorithmischer Sicht als auch die unterschiedlichen Hardwarearchitekturen, die im Zusammenhang dieser Arbeit wichtig sind, behandelt. Dabei orientiert sich der theoretische Teil am Aufbau des Gesamtsystems. Der technologische Grundlagenteil beschäftigt sich mit aktuellen Hardwareplattformen.

Anschließend wird in chapter 5 der Stand der Technik auf dem Gebiet der 3D-Rekonstruktion mit Hilfe der Photogrammetrie behandelt. Dazu werden diejenigen vorhandenen Verfahren zur Korrespondenzanalyse näher vorgestellt, die entweder wegen ihrer Ausgabequalität oder aufgrund ihrer schnellen Berechenbarkeit als geeignet erscheinen. Insbesondere wird auf deren Eignung für ein Echtzeit-Stereo-Vision-System

eingegangen. Zusätzlich werden eine Übersicht und eine Bewertung über existierende kommerzielle Stereo-Vision-Systeme in Abschnitt 5.4 erstellt.

In chapter 6 wird dann ein Konzept für das echtzeitfähige System beschrieben. Das Konzept orientiert sich an den allgemeinen Anforderungen echtzeitfähiger Systeme im Nahbereich, die in section 6.1 erläutert werden.

Anhand spezieller Anforderungen, die in section 7.1 definiert werden, wird im weiteren Verlauf von chapter 7 der in chapter 6 beschriebene Entwurf für den Fall einer mobilen Anwendung konkretisiert. Dabei wird insbesondere ein neuer Algorithmus entwickelt und in chapter 8 vorgestellt, mit dessen Hilfe die Anforderungen aus section 7.1 erfüllt werden können.

Die Diskussion der Ergebnisse des neuen Verfahrens bezüglich der Eigenschaften Skalierbarkeit, Laufzeitverhalten und Güte der Tiefenkarte findet in chapter 9 statt. Zusammenfassung und Ausblick folgen zum Abschluss der Arbeit in chapter 11. Ein Anwendungsbeispiel wird in chapter 10 vorgestellt.

2. Grundlagen

Eine Teilaufgabe dieser Arbeit ist es, die Stereoanalyse in einem Systemzusammenhang zu betrachten, so dass die verschiedenen Faktoren, die einen Einfluss auf die Korrespondenzanalyse haben, mit berücksichtigt werden können. Der Abbildungssprozess der Kamera hat den größten Einfluss auf die Qualität der Stereoanalyse (Abb. 2.1) und befindet sich am Anfang der Verarbeitungskette, so dass die Grundlagen zu dieser Thematik zu Beginn eingeführt werden. Anschließend werden die Grundlagen zur Stereoanalyse behandelt.

2.1. Bildentstehung

Der Prozess der Bildentstehung in einem technischen System ist unter realen Bedingungen fehlerbehaftet. Die einzelnen Bearbeitungsschritte hin zu einem digitalen Bild haben einen unterschiedlich starken Einfluss auf das Endergebnis und werden im nachfolgenden entsprechend dem Informationsfluss näher betrachtet.

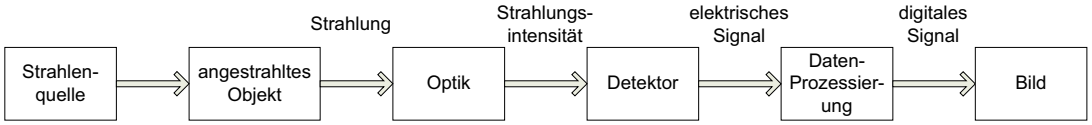


Abbildung 2.1.: Bildentstehung

Optoelektrische Sensoren können nur dann etwas signalisieren, wenn von den Objekten im Sichtfeld Strahlung emittiert oder reflektiert wird, die auf den Detektor trifft. Bevor die Photonen auf den Detektor treffen, passieren diese die Optik,[1]die bereits für eine ganze Reihe verschiedener geometrischer und radiometrischer Abbildungsfehler (Aberrationen) verantwortlich ist.

Optik

Ausschlaggebend für den Grad der optischen Aberration ist die Qualität sowohl der Linsen als auch der Spiegel hinsichtlich des verwendeten Materials und der realisierten Geometrie. Erfahrungsgemäß hat wahlweise die Kissen- oder die Tonnenverzeichnung (Abb. 2.2a und Abb. 2.2b) den stärksten Einfluss.

Die radiale Verzeichnung kann wie in Equation 2.1 durch ein Polynom von beliebiger endlicher Ordnung approximiert werden. Die Wirkung der radialen Verzeichnungen ist

[1]Atmosphärische Eigenschaften seien vernachlässigt.

die, dass sie die abgebildeten Punkte entlang eines Strahls, der seinen Ursprung im Bildhauptpunkt hat, nach innen bzw. nach außen verschieben

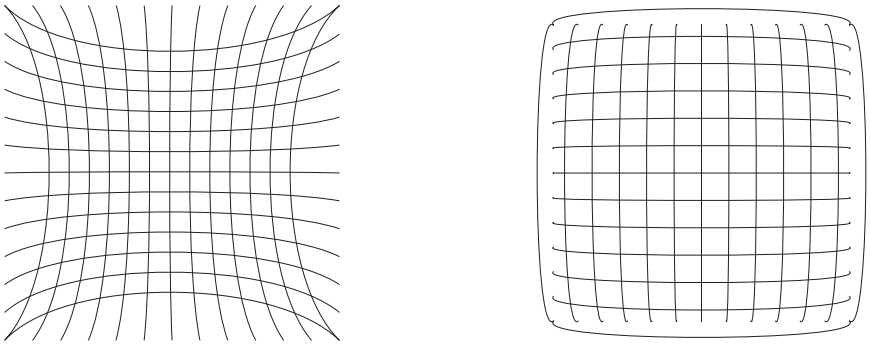
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots) \tag{2.1}$$

mit

- r $= \sqrt{x^2 + y^2}$
- $k_1 \ldots k_n$ = freie Verzeichnungsparameter

Im Vergleich zur radialen Verzeichnung ist die tangentiale relativ gering und häufig vernachlässigbar. Sie resultiert hauptsächlich aus einer Parallelverschiebung der Linsen im Objektiv.

Zur Bestimmung der radialsymmetrischen Verzeichnung sind für weitwinklige Objektive in der Praxis drei freie Parameter ausreichend. Für Standardobjektive ist bereits ein Polynom dritter Ordnung mit einem zu bestimmenden Parameter hinreichend genau. Ähnlich wie die radiale Verzeichnung nimmt die tangentiale mit fortschreitendem Abstand zum Hauptpunkt zu. Insbesondere bei Weitwinkelobjektiven kommt es häufig



(a) Kissenverzeichnung (b) Tonnenverzeichnung

Abbildung 2.2.: Tonnen- und Kissenverzeichnung

vor, dass die Lichtintensität mit zunehmendem Abstand von der Bildmitte abnimmt. Der Grad des Randabfalls wird durch mehrere Faktoren bestimmt, wobei der natürliche Randabfall aufgrund stark schräg einfallender Lichtstrahlen am Objektivrand begründet ist.Verzeichnungsfehler wie sphärische Aberration, Koma, Astigmatismus und Bildfeldwölbung sollen hier nicht betrachtet werden, da sie für Standardobjektive selten auftreten. Kommen Farbkameras zum Einsatz, müssen weitere farbspezifische Abbildungsfehler betrachtet und gegebenenfalls kompensiert werden. Für weitere Ausführungen sei auf die Literatur verwiesen, z.B. [44].

Zur Qualifizierung des geometrischen Auflösungsvermögens unterschiedlicher Optiken werden in der Praxis die Kennlinien der sogenannten Modulationsübertragungsfunkti-

on (MTF, *modulation transfer function*) miteinander verglichen. Dabei wird das Objektiv als ein Filter betrachtet. Das Ergebnis aus der Faltung der Übergangsfunktion der Optik, auch Punktantwort (PSF, *point spread function*) PSF(x) mit der Ortsfunktion $o(x)$ einer Punktquelle wird fourier-transformiert und als das Frequenzspektrum des Bildes $G(x)$ bezeichnet.

$$\begin{aligned} g(x) &= \mathrm{PSF}(x) * o(x) \\ &\circ\!\!-\!\!\bullet \\ G(k) &= \mathrm{OTF}(k) \cdot O(k) \end{aligned} \tag{2.2}$$

Der Betrag der Division von $G(x)$ und $O(k)$ ist die MTF:

$$\left|\frac{G(k)}{O(k)}\right| = \mathrm{MTF}(k)$$

In einer zusammenhängenden Betrachtung der Optik und des Detektors, können beide Kennlinien miteinander multipliziert werden:

$$\mathrm{MTF}_{\mathrm{gesamt}} = \mathrm{MTF}_{\mathrm{Optik}} \cdot \mathrm{MTF}_{\mathrm{Detektor}}$$

Detektoren

Im Bereich des sichtbaren Lichts haben sich Bauelemente auf der Basis von zwei Detektortechnologien etabliert. Das sind zum einen die Charged-Coupled-Devices (CCD) und zum anderen die Complementary-Metal-Oxide-Semiconductor-(CMOS)-Bauelemente. Beim Vergleich der Technologien spielen die spektrale Empfindlichkeit, das SNR und die Abweichungen von den Detektorlinearitäten[2](DSNU, PRNU) eine wichtige Rolle. Ist zum Beispiel das SNR zu gering, ist ein Matching korrespondierender Punkte auf der Basis von Grauwertunterschieden nicht mehr möglich bzw. es ergeben sich Mehrdeutigkeiten, die vom Matching-Algorithmus nicht mehr aufgelöst werden können.

Die folgende Aufzählung nennt weitere Kennzeichen zur Beschreibung der Detektoren:

- Die **Quanteneffizienz** oder auch Quantenausbeute ist ein Maß für den prozentualen Anteil der einfallenden Photonen, die in elektrische Ladungen umgewandelt werden. Sie ist bestimmt durch das Material und die Struktur des Detektors.
- Die **Sensitivität** eines Sensor gibt an, wieviel Photonen nötig sind, um ein Signal zu erzeugen. Sie ist ein Verhältnis aus SNR und Quanteneffizienz.
- Die **Detektorgröße** (aktive Fläche) errechnet sich aus der Pixelanzahl und der technisch bedingten Abstände zw. den Pixeln.

[2]DSNU, *dark signal non uniformity*; PRNU, *photo response non uniformity*

- Bildzeilen können **progressiv** oder **interlaced** ausgelesen werden; letzteres ist von der Fernsehtechnik übernommen.
- Der **Elektronentransfer** in CCD-Sensoren kann auf unterschiedliche Arten implementiert sein und hat ebenfalls Einfluss auf die Qualität der Abbildung (interline, frame, full-frame, frame-interline)
- Der **Shutter** kann für das vollständige Bild, zeilenweise (rollend) oder partiell (blockweise) ausgeführt werden.
- Die **Auslesegeschwindigkeit** von CCDs ist abhängig vom Shutter und dem Elektronentransfer.
- Die **Sättigungszahl** (engl. full well capacity) gibt an, wieviel Elektronen ein Pixel aufnehmen kann, bevor diese auf benachbarte Potentiale übertreten (Blooming). Je größer dieser Wert ist, desto größer ist auch die Dynamik des Detektors. Die Sättigungszahl hängt von der Größe des Pixels ab. Der Blooming-Effekt kann durch spezielle Gatter minimiert werden, die allerdings die Lichtempfindlichkeit negativ beeinflussen.
- Das **radiometrische Auflösungsvermögen** wird durch die Dynamik des Detektors und durch das Auflösungsvermögen des AD-Wandlers bestimmt.
- Die **Dynamik** oder auch das Signal-Rausch-Verhältnis (SNR) errechnet sich aus dem Quotienten der Sättigungszahl und dem minimalen Rauschanteil.

Das SNR ist neben der MTF gemeinhin das wichtigste Vergleichskriterium für die unterschiedlichen Sensoren. Es ergibt sich aus dem Verhältnis von Nutzsignal zu Rauschsignal. Zum Beispiel kann das SNR für CCD-Sensoren als ein Verhältnis aus der Anzahl der Signalelektronen zu Rauschelektronen definiert werden. In dem Fall entspricht das Nutzsignal der Anzahl der Signalelektronen N_S und das Rauschsignal der Anzahl der Rauschelektronen N_R. Die Anzahl der Signalelektronen ist definiert als:

$$N_S = \frac{\Phi}{h \cdot \nu} \cdot t \cdot A \cdot \eta \tag{2.3}$$

mit

- N_S = Anzahl der Signalelektronen, die von dem Detektor erfasst werden
- Φ = Lichtleistung [W / m^2]
- $h \cdot \nu$ = Photonenenergie [Ws]
- A = Pixelgröße [m^2]
- η = Quanteneffektivität des Detektors

Das Rauschsignal wird durch eine Reihe unterschiedlicher Rauschquellen verursacht (Abb. 2.3) und ist von vielen Faktoren abhängig. Eine nicht vermeidbare Rausch-

quelle ist das Photonenrauschen $N_{Photonen} = \sqrt{N_S}$, das eine untere Schranke für das Rauschverhalten in klassischen Aufnahmesystemen darstellt.

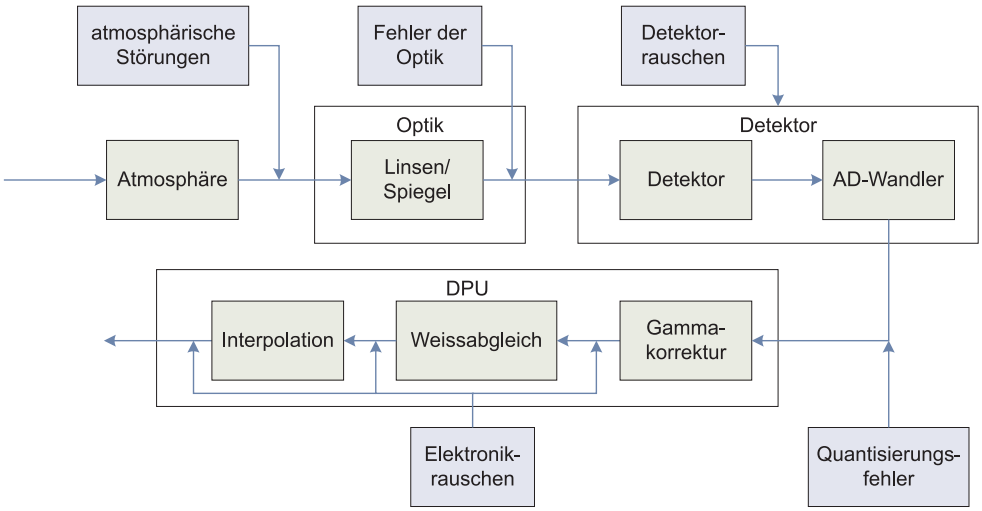


Abbildung 2.3.: Fehlerquellen einer Kamera

Unter Detektorrauschen N_{Detektor} werden die Begriffe Dunkelrauschen, Transferrauschen, thermisches Strukturrauschen und Verstärkerrauschen zusammengefasst, wobei das thermische Strukturrauschen dominiert und sich besonders bei Aufnahmen mit geringem Kontrast und langer Belichtungszeit bemerkbar macht. Photonenrauschen wird üblicherweise als ein Poisson-Prozess über der Zeit modelliert und ist von der einfallenden Lichtmenge abhängig. Aufgrund der Eigenschaften eines Poissonprozesses gilt, dass der Erwartungswert gleich der Varianz ist. Zudem kann die Poisson-Verteilung durch eine Normal-Verteilung mit gleichem Mittelwert und Varianz approximiert werden. Zusammengefasst ergibt sich die Standardabweichung des Rauschsignals als:

$$\sigma_{\text{Rauschsignal}} = \sqrt{N_{Photonen}^2 + N_{\text{Detektor}}^2} \tag{2.4}$$

Aus den Gleichungen 2.3 und (2.4) folgt das SNR für einen abbildenden Sensor im sichtbaren Wellenlängenbereich:

$$\text{SNR} = \frac{\mu_{Nutzsignal}}{\sigma_{Rauschsignal}} = \frac{N}{\sqrt{N_{Photonen}^2 + N_{\text{Detektor}}^2}} \tag{2.5}$$

Unter verschiedenen Aufnahmebedingungen kann Equation 2.5 vereinfacht werden. Für starke Lichtverhältnisse kann das SNR durch

$$\text{SNR} \approx \frac{N_S}{\sqrt{\left(\sqrt{N_S}\right)^2}} \tag{2.6}$$

$$= \sqrt{N_S} \tag{2.7}$$

approximiert werden. Im direkten Vergleich unterscheiden sich CCD- und CMOS-Sensoren vorrangig hinsichtlich der Lichtempfindlichkeit. Die lichtempfindliche Fläche ist bei CMOS-Sensoren aufgrund der zusätzlichen Elektronik kleiner, wodurch die Gesamtempfindlichkeit leidet. Des Weiteren weisen CCDs hinsichtlich der Gleichförmigkeiten (DSNU und PRNU) geringere Abweichungen auf.

Datenprozessierung

Die analogen Ausgangssignale des Detektors werden durch einen Analog-Digital-Converter (ADC) digitalisiert, der durch eine Reihe von Parametern charakterisiert wird. Dazu zählen:

- das Auflösungsvermögen
- die Abtastrate
- und das Rauschverhalten.

Grundsätzlich gilt, dass eine langsame Abtastung eine höhere Auflösung bewirkt und umgekehrt. Die notwendige minimale theoretische Auflösung eines ADC in Relation zum Dynamikumfang eines optischen Sensors kann mit

$$\text{Auflösung bits} = \frac{\text{SNR } -1,76 \text{ dB}}{6,02 \text{ dB}}$$

berechnet werden [84]. Das SNR für CCD-Kameras beträgt in der Regel zwischen 60 dB und 74 dB, so dass die Wandlung das analoge Signal mit mindestens 10 bis 12 bit auflösen sollte.

Um die Ausgabe des Sensors zu kalibrieren, werden radiometrische Korrekturen wie Gain- und Weissabgleich in der Datenprozessierungseinheit (DPU, *data processing unit*) durchgeführt. Diese Korrekturen und die Digitalisierung selbst stellen weitere Rauschquellen dar. Des weiteren werden Inhomogenitäten und defekte Pixel kompensiert sowie PRNU- und DSNU-Korrekturen von der DPU durchgeführt.

2.2. Geometrische Kamerakalibrierung

Für den Nahbereich hat sich in der Photogrammetrie die zentralperspektivische Projektion als Modell für die Geometrie der Bildentstehung etabliert. Die perspektivische Projektion ist eine im projektiven Raum definierte lineare Abbildung. Je nachdem wo sich zwei gleich große Objekte in der Tiefe befinden, werden sie unterschiedlich groß auf die Bildebene projiziert. Die Lochkamera (bzw. das Lochkameramodell) (Abb. 2.4) ist die einfachste Realisierung einer perspektivischen Projektion und für die meisten photogrammetrischen Anwendungen eine ausreichende Generalisierung.

Zu den Eigenschaften projektiver Geometrie gehört:

- Punkte werden im projektiven Raum durch homogene Koordinaten ausgedrückt.

- Homogene Koordinaten $\mathbb{P}^n$ werden als (n+1)-dimensionale Vektoren formuliert.
- Homogene Koordinaten sind gegenüber Skalierung invariant, was anhand der Definition projektiver Räume deutlich wird.
- Der projektive (reelle) Raum $\mathbb{P}^n$ wird definiert als $\mathbb{P}^n := \mathbb{R}^{n+1} \setminus \{0\}$ und der Aquivalenzrelation $\sim$. Letzteres wird definiert als $x \sim y \Leftrightarrow \exists \lambda \in \mathbb{R} \setminus \{0\}$ für die gilt $x = \lambda y$.
- Im Vergleich zum euklidischen kann im projektiven Raum mit Punkten im Unendlichen gearbeitet werden.
- Matrixmultiplikationen und -additionen lassen sich in homogener Form kompakt ausführen.

Die Grundlage projektiver Abbildungen bilden homogene Koordinaten $p = (x, y, z)^T \in \mathbb{P}^2$, die in euklidische Form durch

$$f : p = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow q = \begin{pmatrix} x/z \\ y/z \end{pmatrix}$$

mit $(x, y, z)^T \sim (x/z, y/z, 1)^T$ gebracht werden können. Umgekehrt kann aus euklidischen Koordinaten die homogene Form durch

$$f : q = \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

mit

- p = Punkt im projektiven Raum $\mathbb{P}^2$
- q = Punkt im euklidischen Raum $\mathbb{R}^2$

aufgestellt werden.

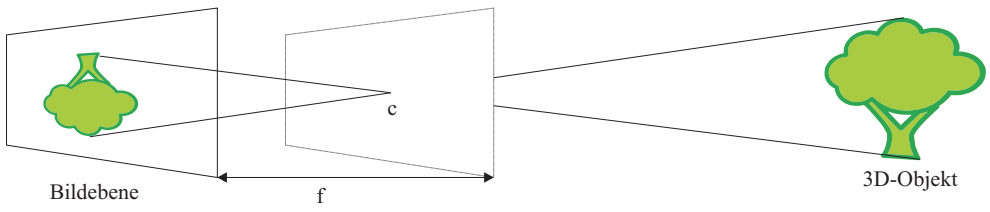


Abbildung 2.4.: Das Lochkameramodell

Im Fall des Lochkameramodells findet eine projektive Transformation $\mathbb{P}$ der Objekt-

punkte $P \in \mathbb{P}^3$ auf einen Bildpunkt $p \in \mathbb{P}^2$ statt und ist definiert durch:

$$p = \mathbb{P} \cdot P \tag{2.8}$$
$$= K[R|t] \cdot P \tag{2.9}$$

mit

- $R = 3 \times 3$ Rotationsmatrix
- $t =$ Translationsvektor

Die homogene 3×4 Matrix $[R|t]$ beinhaltet die Parameter der äußeren Orientierung und stellt eine Verbindung zwischen dem Kamerakoordinatensystem und dem übergeordneten Weltkoordinatensystem her. Die Matrix K dient der Transformation vom Kamerakoordinaten- auf das Bildkoordinatensystem und beschreibt die Parameter der inneren Orientierung [43].

Innere Orientierung

Alle Raumpunkte P lassen sich über ein optisches Zentrum bzw. Projektionszentrum c auf die Bildebene abbilden. Der Abstand von der Bildebene zu dem Projektionszentrum c wird als Kamerakonstante f bezeichnet. Als Bildhauptpunkt H ist der Punkt auf der Bildebene definiert, an dem die optische Achse die Bildebene als Lotfußpunkt trifft. Der Koordinatenursprung des Bildkoordinatensystems liegt per Definition in der oberen linken Bildecke. Der Koordinatenursprung des Kamerakoordinatensystems befindet sich wiederum im Projektionszentrum c. Die Werte für die Verschiebung des Bildhauptpunktes H vom Koordinatenursprung des Bildes u_0, v_0 in Pixel und die Kammerakonstante f bilden die Parameter der sog. inneren Orientierung K und beschreiben damit die geometrischen Verhältnisse innerhalb der Kamera (Abb. 2.5).

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Die Transformation eines Raumpunktes P in das Bildkoordinatensystem lautet in homogener Matrixschreibweise:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K[I|0] \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.11}$$

mit

- $I =$ Einheitsmatrix

Äußere Orientierung

Die Parameter der äußeren Orientierung beschreiben zum einen die Lage des Projektionszentrums der Kamera in einem übergeordneten Koordinatensystem und zum anderen die Ausrichtung des Kamerakoordinatensystems in diesem übergeordneten System. Die Transformation eines Punktes $p \in \mathbb{P}^3$ aus dem Weltkoordinatensystem in das Kamerakoordinatensystem ist definiert als

$$p_K = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.12}$$

mit

- $p_K \in \mathbb{P}^3$ = Punkt im Kamerakoordinatensystem
- $R = 3 \times 3$ Rotationsmatrix, die die Orientierung des Kamerakoordinatensystems im übergeordneten System beschreibt
- C = Ursprung des Kamerakoordinatensystems im Weltkoordinatensystem

In Stereoanordnungen ist es sinnvoll, das gemeinsame Weltkoordinatensystem identisch zum Koordinatensystem der ersten Kamera zu definieren, sodass Rotation und Translation vom zweiten Kamerakoordinatensystem relativ zu diesem sind. Dadurch vereinfacht sich Equation 2.12

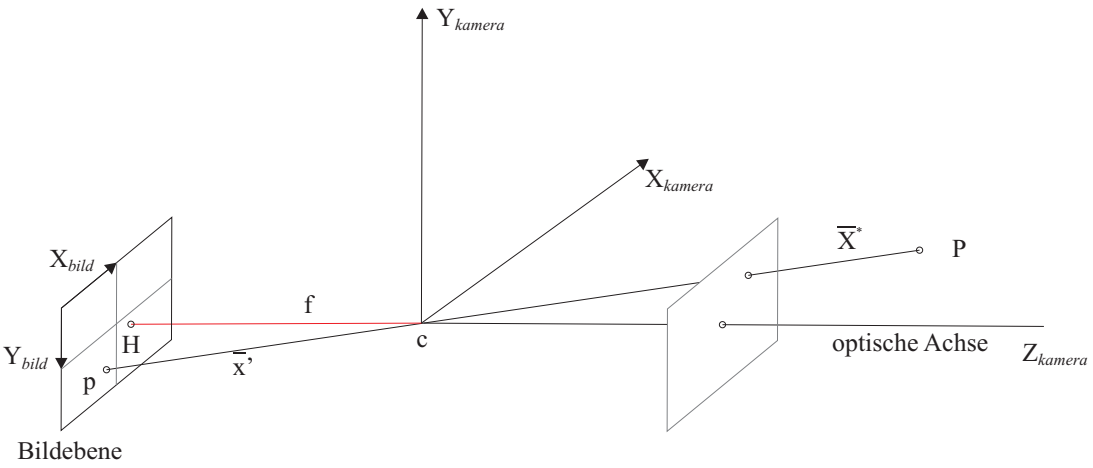


Abbildung 2.5.: Zusammenhang zwischen Bild- und Kamerakoordinaten [77]

$$p_K = \left[R \mid t\right] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.13}$$

mit

- t = Translation des Projektionszentrums der beiden Kameras zueinander

Durch Hinzunahme der Parameter der inneren Orientierung ist eine vollständige Beschreibung zwischen Objekt- und Bildpunkten hergestellt, sodass gilt:

$$p = K[R|t]P \tag{2.14}$$

und auf die Equation 2.9 in der Einführung zurück führt. Für eine photogrammetrische Bestimmung des 3D-Objektpunktes P müssen somit die Parameter der inneren und äußeren Orientierung bekannt sein.

Erst durch Triangulation mit einem zweiten Raumstrahl (bzw. mit einer bekannten Ebene), wie in Abb. 2.6 gezeigt, kann zusätzlich zu der Richtung des Raumpunktes P auch dessen Tiefe berechnet und der anfänglich unbestimmte Maßstabsfaktor λ aufgelöst werden. Die Grundlage dazu bildet die Epipolargeometrie in deren Mittelpunkt die Equation 2.17 steht. Korrespondierende Punkte, die in mindestens zwei Bildern

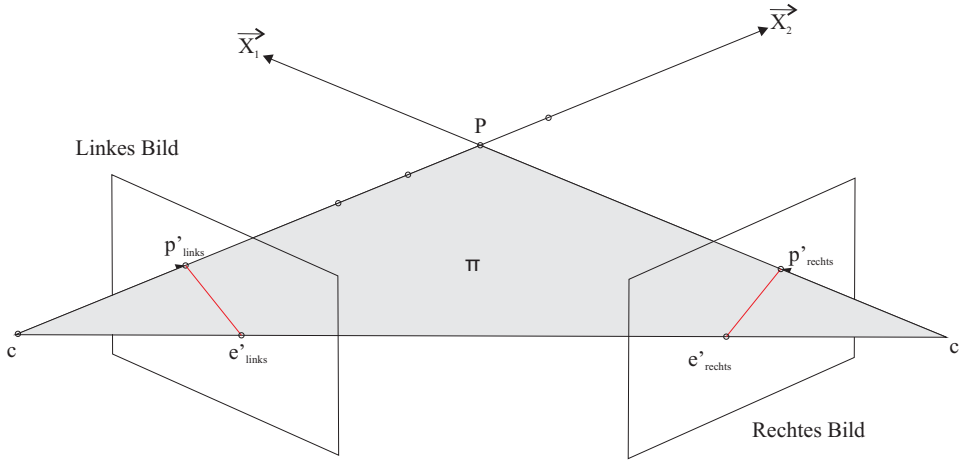


Abbildung 2.6.: Epipolargeometrie für einen konvergenten Stereoaufbau [77]

gesehen werden, müssen sich im Idealfall auf einer zwischen den Projektionszentren und den Punkten gespannten Ebene Π, der Epipolarebene, befinden. Die Schnittgerade zwischen der Epipolarebene und der Bildebene ist die Epipolarlinie. Im Anwendungsfall kann es vorkommen, dass die beiden Strahlen $\overrightarrow{X_1}$ und $\overrightarrow{X_2}$ sich nicht im Raum schneiden, sondern sich leicht windschief zueinander verhalten.

Die Epipolargeometrie hat entscheidenden Einfluss auf die Komplexität der Stereoanalyse. Bei bekannter relativer Orientierung der Stereobildaufnahmen zueinander wird der Suchraum für korrespondierende Punktepaare auf die Epipolarlinie eingeschränkt. Im allgemeinen Fall der Stereobildauswertung, wie er in der Abb. 2.6 dargestellt ist, erfolgt die Suche nach homologen Punkten entlang einer diagonal in der Bildebene liegenden Epipolarlinie. Für unkorrigierte Luftbildaufnahmen mit Zeilenkameras kann die Epipolarlinie zusätzlich gekrümmt sein. Durch eine Transformation der Bilder auf eine gemeinsame Ebene können die Mehrbildaufnahmen aus dem allgemeinen in den Stereonormalfall (Abb. 2.7) überführt werden. Die Epipolarlinie verläuft dann entlang der x-Achse, wodurch erhebliche Vorteile bei der Verarbeitung durch den linearen Pixelzugriff zu erwarten sind.

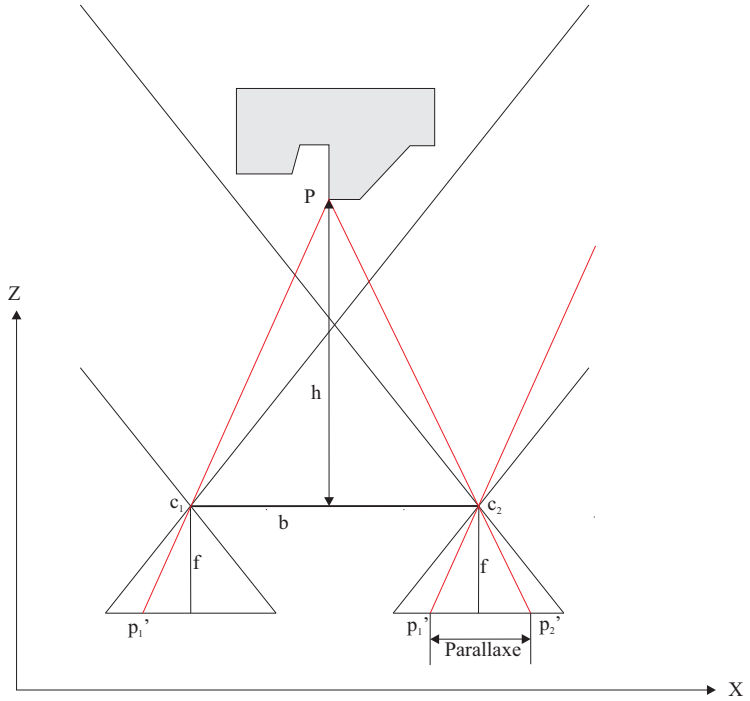


Abbildung 2.7.: Der Stereonormalfall [77]

Geometrische Auflösung

Zur Koordinatenberechnung der Z-Komponente im Stereonormalfall genügen die einfachen Verhältnisse

$$Z = h = \frac{b \cdot f}{x'_1 - x'_2} \tag{2.15}$$

mit

- h = Tiefe des Objektpunktes im Kamerakoordinatensystem

- b = Basisabstand der beiden Projektionszentren c_1 und c_2
- x'_1 = x-Komponente von Punkt p'_1
- x'_2 = x-Komponente von Punkt p'_2

Der Betrag $|x'_1 - x'_2|$ wird als x-Parallaxe oder Disparität bezeichnet und in Pixeln angegeben. Durch die Abbildung eines Raumes auf einen diskreten Sensor wird der Begriff der Tiefenauflösung geprägt und gibt ein quadratisches Verhältnis zwischen der Tiefenauflösung R und der Entfernung h vom Projektionszentrum c zum Objekt wieder (Abb. 2.8).

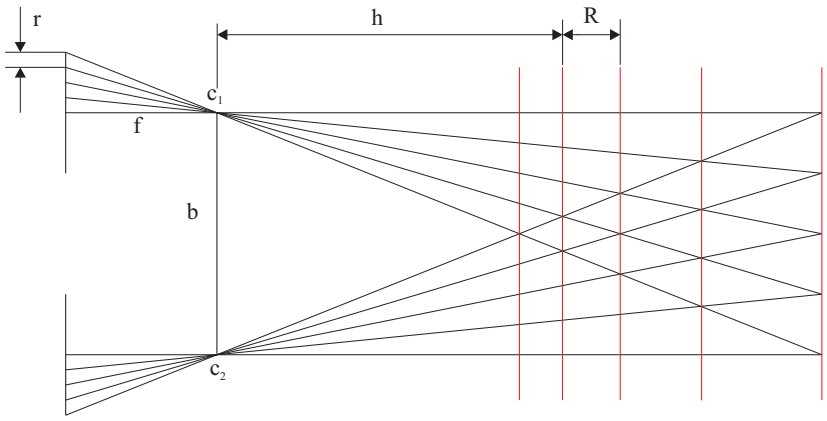


Abbildung 2.8.: Stereotiefenauflösung

Equation 2.16 stellt diesen Zusammenhang mathematisch dar, während in Abb. 2.9 der quadratische Anstieg exemplarisch für f=4,6 mm aufgetragen ist.

$$R = \frac{r \cdot h^2}{f \cdot b - r \cdot h} \tag{2.16}$$

mit

- R = Tiefenauflösung
- r = Größe eines Pixels

Eine große Basislänge ermöglicht demnach eine detailliertere Tiefenauflösung. Bei diesem Ansatz wirken sich jedoch die größeren Abschattungsbereiche und die stärker ins Gewicht fallenden perspektivischen Unterschiede nachteilig aus. Um dennoch bei Verringerung des Basisabstands eine gleich bleibende Tiefenauflösung zu gewährleisten, muss die Sub-Pixelgenauigkeit der Korrespondenzanalyse präzisiert werden. Methoden dazu werden in Abb. 2.4 näher erläutert. Bei bekannter innerer Kalibration, rektifizierten Bildpaaren und errechneter Parallaxe kann auf einfachem Wege die Tiefe rekonstruiert werden. Im section A.2 des Anhangs findet sich eine detaillierte Herleitung.

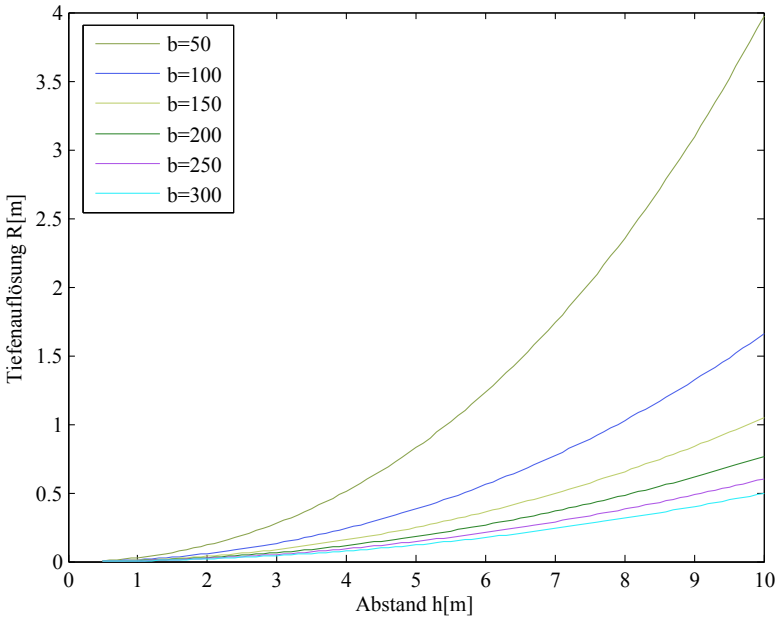


Abbildung 2.9.: Verhältnis zwischen Tiefenauflösung R, Abstand zum Objekt h und Basislänge b in mm. Die angenomme Pixelgröße beträgt 6,4 µm.

Rektifizierung von Stereoaufnahmen

Rechentechnisch kann die Suche nach korrespondierenden Punkten erheblich vereinfacht werden, wenn alle Epipolarlinien eines Bildes entlang der X-Achse verlaufen. Durch Rektifizierung können die Eingangsbildpaare in den Stereonormalfall transformiert werden. Ausgangspunkt ist die Koplanaritätsbedingung aus Equation 2.17, die die korrespondierenden Bildpunkte $p_1 \in \mathbb{P}^2$ und $p_2 \in \mathbb{P}^2$ in einen mathematischen Zusammenhang bringt:

$$p_1^T \cdot F \cdot p_2 = 0 \tag{2.17}$$

F wird als Fundamentalmatrix bezeichnet und enthält die Parameter der inneren und der relativen Orientierung der beiden Kameras. F ist eine homogene, nicht invertierbare 3×3-Matrix mit $\text{rang}(F) = 2$. Bei bekannter relativer und innerer Orientierung beider Kameras kann F durch

$$F = K_1^{-T} \cdot E \cdot K_2^{-1} \tag{2.18}$$

$$E = [t] \times R \tag{2.19}$$

- E = essentielle Matrix (relative Orientierung)
- K_1, K_2 = Parameter der inneren Orientierung der beiden Kameras

direkt bestimmt werden. Im unkalibrierten Fall[3] kann F direkt durch die Bestimmung weniger Punktkorrespondenzen ermittelt werden. Bekannte Verfahren sind der 7- oder 8-Punkt-Algorithmus [43].

Zur Überführung des Systems in den Stereonormalfall müssen die Bildebenen beider Kameras auf eine gemeinsame Ebene gebracht werden. Dazu wird eine Kamera um dessen Projektionszentrum rotiert. Um die Epipolarlinien parallel zur x-Achse auszurichten, müssen möglicherweise beide Kameras rotiert werden. Anschließend muss eventuell die Skalierung der Bilder angepasst werden. Die notwendigen Projektionsmatrizen H_1 und H_2 für die Transformation der Bildpaare im projektiven Raum können aus F heraus bestimmt werden. Die Punkte p' und p, sowie die Matrizen H_1 und H_2 sind in homogener Form gegeben:

$$p'_1 = H_1 \cdot p_1 \tag{2.20}$$
$$p'_2 = H_2 \cdot p_2 \tag{2.21}$$

Für rektifizierte Bildpaare gelten eine Reihe von Bedingungen:

- Die Epipole liegen im Unendlichen
- und alle Epipolarlinien verlaufen parallel zu x-Achse.

Durch Ersetzung korrespondierender Punkte p_1 und p_2 mit den jeweiligen Termen aus den Gleichungen (2.20) und (2.21) in Equation 2.17:

$$p_2'^T \cdot F' \cdot p'_1 = 0 \tag{2.22}$$
$$p_2^T \cdot H_2^T \cdot F' \cdot H_1 \cdot p_1 = 0 \tag{2.23}$$

kann Equation 2.24 formuliert werden als:

$$F = H_2'^T \cdot F' \cdot H_1 \tag{2.24}$$

Die Fundamentalmatrix für rektifizierte Bilder F' ist definiert als

$$F' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.25}$$

Zur Berechnung von H_1 und H_2 werden diese üblicherweise in einfachere Transformationen zerlegt, wobei aufgrund der Mehrdeutigkeit von Equation 2.24 mehrere Lösungen existieren. Die bekanntesten Lösungsverfahren von C. Loop et al. [76], A. Fusiello et al. [29], R. Hartley et al. [43] und J. Mallon et al. [79] transformieren zunächst die Epipole beider Bilder ins Unendliche. Anschließend werden die eventuell unterschiedlichen Brennweiten durch Anpassung des Skalierungsfaktors korrigiert, so dass die Epipolarlinien auf einer Geraden liegen.

[3] Weder die innere noch die relative Orientierung sind bekannt.

2.3. Radiometrische Korrekturen

Jeder optische Sensor hat charakteristische Eigenschaften. Das betrifft unter anderem die spektrale Empfindlichkeit, das statische Festmusterrauschen, die Anzahl defekter Pixel und das Gesamtkamerarauschen. Statische Fehler wie zum Beispiel PRNU sowie DSNU werden üblicherweise schon werksseitig bestimmt und durch die Ressourcen der Kamera korrigiert. Dynamische Effekte (Alterungsprozesse) können gegebenenfalls nachträglich kompensiert werden.

Radiometrische Unterschiede, die durch die Aufnahmesituation entstanden sind, werden ebenfalls im Nachgang berücksichtigt. Insbesondere für Luft- und Satellitenbilder gilt, dass Bildverbände zu verschiedenen Zeitpunkten unter veränderten Umweltbedingungen entstehen und diese starke radiometrische Unterschiede aufweisen können. Das gleiche gilt für Stereoaufnahmen mit zwei unterschiedlichen Kameratypen.

Bilder, die unter realistischen Bedingungen aufgenommen worden sind, besitzen relativ häufig ein schlechtes SNR, so dass zu dessen Verbesserung Filter auf das Eingangsbild angewendet werden.[4] Nur bedingt geeignete Filter sind zum Beispiel der Mean-Filter oder der Laplacian-of-Gaussian (LoG), die eine mehr oder minder ausgeprägte Verwischung der Kanten bewirken, was sich im Disparitätenbild widerspiegelt. Der häufig verwendete bilaterale Filter [106] wirkt sich dagegen kantenerhaltend aus.

2.4. Korrespondenzanalyse

Die Korrespondenzanalyse ist die zentrale Thematik für das räumliche Sehen und beruht darauf, korrespondierende Punkte in zeitlich und/oder räumlich versetzten Bildaufnahmen wiederzufinden. Die Suche nach korrespondierenden Punkten geht dabei von der Annahme aus, dass deren Merkmale in beiden Bildern gleich bzw. zumindest ähnlich sind. Als Vergleichsgrundlage werden häufig die Grauwerte der Pixel (Intensität) oder die Ableitungen erster und zweiter Ordnung der Bilder verwendet.

Fehlerursachen

Eine Korrespondenzanalyse, die ausschließlich die Intensität eines Pixels betrachtet, ist aufgrund der in der Praxis immer enthaltenden Mehrdeutigkeiten so gut wie unmöglich. Weiterhin können mit Stereokameraaufbauten in verdeckten Bereichen der Szene überhaupt keine homologen Punkte gefunden werden. Durch drei oder mehr Perspektiven können die verdeckten Bereiche minimiert werden, jedoch gilt allgemein, dass das Korrespondenzproblem nicht eindeutig lösbar ist. Das hat zum einen geometrische und zum anderen radiometrische Ursachen, die nachfolgend in der Reihenfolge ihrer Bedeutung aufgelistet sind.

Geometrische Ursachen

- Überdeckungen: Es existieren keine homologen Punktepaare in den Bildern.

- Unterschiedliche Ansichten des Objekts können nicht mehr aufgelöst werden.
- Perspektivische Verzerrungen (foreshortening) werden durch die perspektivische Abbildung verursacht und bewirken eine Verkürzung zum Beispiel von Linien, die aus unterschiedlichen Winkeln betrachtet werden.

Radiometrische Ursachen

- Ein geringes SNR verursacht Mehrdeutigkeiten.
- Transparente Objekte verursachen Mehrdeutigkeiten.
- Unterschiedliche Beleuchtungverhältnisse können nicht mehr aufgelöst werden.
- Bei Mehrdeutigkeiten aufgrund von schlecht oder untexturierten Bereichen existieren mehrere potentiell korrespondierende Punktepaare.

Gegenmaßnahmen

Durch geeignete Gegenmaßnahmen können diese Störfaktoren bereits bei der Bildentstehung minimiert werden. Dazu zählt typischerweise eine Erhöhung des Aufwandes bzgl. der Beleuchtung, der Optik und der Sensoren. Neben dem Aufnahmeverhältnis spielt die Laufzeit für die Korrespondenzanalyse eine wichtige Rolle. Es lassen sich eine Reihe von Nebenbedingungen bzw. Prinzipien formulieren, die dazu führen, dass der Suchraum für die Korrespondenzanalyse eingeschränkt werden kann. Sowohl Vorkenntnis der Szenerie kann dafür genutzt werden als auch Information über die Lage der Aufnahmesysteme.

- Die mitunter wichtigste Annahme ist das Prinzip der **Epipolargeometrie**, für die im Stereonormalfall gilt, dass die Epipolarlinie parallel zur x-Achse des Bildes verläuft (Equation 2.2) und durch Vorverarbeitungsschritte sichergestellt wird.
- Mit dem Begriff **Local Smoothness** verbindet sich die Annahme, dass die Disparitätsunterschiede in einer kleinen Umgebung moderat sind. Für diese Nebenbedingung müssen jedoch besondere Maßnahmen an Objektkanten getroffen werden, da hier die Annahme einer lokalen Ähnlichkeit nicht gilt. Das bedeutet, dass die Einschränkung auf die lokale Glattheit, ohne zu wissen, wo sich Objektgrenzen befinden, zwangsläufig zu Fehlern an eben jenen Kanten führt. Dies gilt insbesondere an Stellen, wo Disparitätssprünge zu erwarten sind. Ein Vorwissen über die Szenengestaltung bzw. über die im Bild abgebildeten Objekte ist demnach notwendig, um die Nachbarschaftsbereiche exakt einzugrenzen. Durch Segmentierung nach Objekten im Bild können diese Bereiche bestimmt werden. Die Rekonstruktion der Tiefe durch die Korrespondenzanalyse ist jedoch in gewisser Hinsicht ein Vorverarbeitungsschritt, um Objekte im Raum zu identifizieren, so dass hier prinzipiell iterativ vorgegangen werden muss. Eine Verbesserung der Segmentierung stützt die Korrespondenzanalyse, deren verbessertes Ergebnis wiederum der Segmentierung zu Gute kommt.

- Die Einschränkung auf **frontal-parallele Szenen** hat zur Folge, dass fließende Tiefenübergänge nicht modelliert werden können bzw. zu starken Aliasing-Effekten führten.
- Das **Eindeutigkeitsprinzip** besagt, dass es für einen Punkt im Referenzbild genau einen oder keinen korrespondierenden Punkt in dem zu matchenden Bild gibt. Diese Annahme gilt zum Beispiel nicht für transparente Objekte. Des weiteren wird häufig angenommen, dass sich geometrische Formen in den Bildpaaren ähnlich sind. Kanten bleiben als Kanten erkennbar.
- Eine weitere typische Einschränkung ist das **Ordnungsprinzip**. Das Ordnungsprinzip besagt, dass die Anordnung bzw. Reihenfolge zweier Pixel zwischen zwei Ansichten identisch ist. Abb. 2.10 zeigt einen relativ typischen Stereofall, in dem das Ordnungsprinzip nicht eingehalten wird. Algorithmen, die das Ordnungsprinzip erzwingen, liefern an dieser Stelle falsche Punktkorrespondenzen.
- Häufig vorkommende Modelleinschränkungen bzgl. der Szene betreffen die Objekttransparenz, die Objektform und die Objekttexturierung.

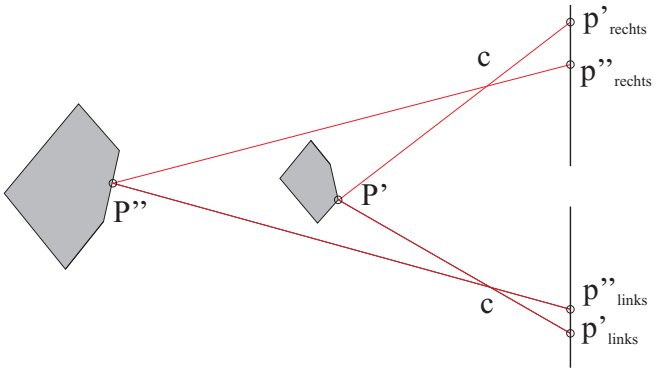


Abbildung 2.10.: Verletzung des Ordnungsprinzips

Klassifizierung der Verfahren

Hinsichtlich des Endergebnisses wird zwischen dünn besetzten (*sparse*) und dichten Tiefenkarten (*dense*) unterschieden. Typischerweise werden 2D-Tiefen- oder auch Disparitätskarten erzeugt, die zusammen mit einem Referenzbild weiterverarbeitet werden. Prinzipiell wird bei der Mehrzahl der Analyseverfahren durch den direkten Vergleich der Grauwerte versucht, homologe Punkte zu identifizieren. Andere wiederum arbeiten indirekt, indem sie das Eingangsbild vorab in eine andere Darstellung transformieren, um so abstraktere Objekte wie zum Beispiel Kanten oder Merkmalspunkte einander zuzuordnen.

Abbildung 2.11 gibt einen Überblick über die unterschiedlichen algorithmischen Herangehensweisen. Globale Verfahren berücksichtigen dabei die gesamte zur Verfügung gestellte Bildinformation. Disparitäts- und Tiefenkarten unterscheiden sich dahinge-

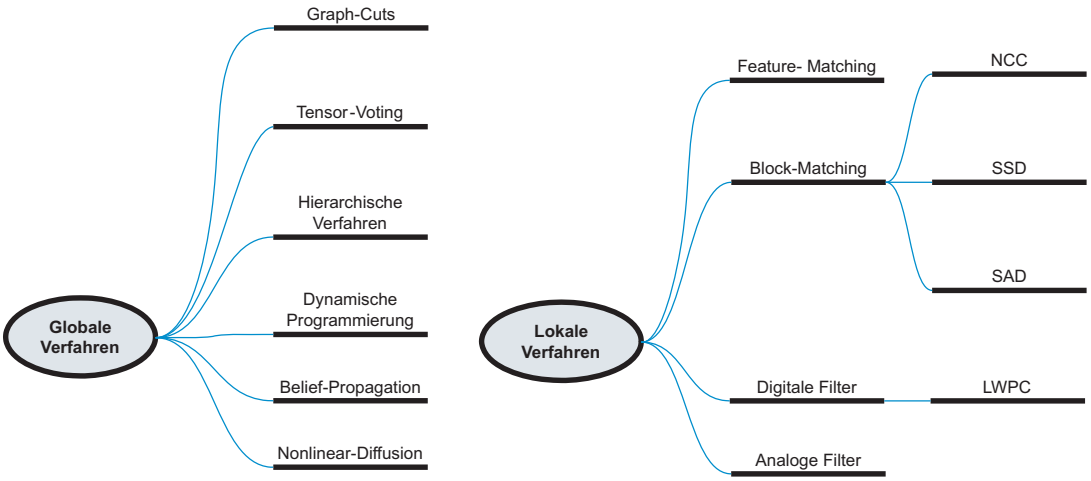


Abbildung 2.11.: Lokale und globale Methoden zur Gewinnung von Tiefeninformation

hend, dass erstere Auskunft über den Abstand der Objekte in Pixeln geben, während Tiefenkarten aus diesen abgeleitet werden und die Entfernung in Metern angeben (section A.2).

Kostenfunktionen

Üblicherweise besteht der erste Schritt darin, eine geeignete Kostenfunktion aufzustellen, auf deren Grundlage eine Bewertung für die Zuordnung der einzelnen Pixel einer Parallaxe durchgeführt werden kann. Daran knüpft eine Phase an, in der die Kosten in einer definierten Art und Weise zusammengeführt werden. Anschließend werden anhand einer Strategie die Disparitäten ermittelt und in einer abschließenden Phase nachbearbeitet bzw. verfeinert. Als eine mögliche Gliederung der Matchingverfahren hinsichtlich ihrer Struktur hat sich die von D. Scharstein et al. [97] eingeführte Taxonomie durchgesetzt, die hier aufgegriffen wird.

Sparse-Stereo-Matcher basieren darauf, markante Merkmale bzw. Objekte wie zum Beispiel Kanten oder einfach zu beschreibende geometrische Formen wiederzufinden. Zur Merkmalsextraktion eignen sich typischerweise Kanten- oder Interestoperatoren oder pyramidale Ansätze. Aufgrund der üblicherweise sehr günstigen Laufzeiteigenschaften werden in echtzeitkritischen Anwendungen Sparse-Stereo-Matcher bevorzugt. Prinzipiell ist es möglich, durch Interpolation aus der dünnbesetzten Tiefenkarte eine dichte Variante zu gewinnen. Der ursprünglichen Ressourceneinsparung stehen hierbei jedoch der zusätzliche Rechenaufwand und das relativ schlechte Endergebnis gegenüber.

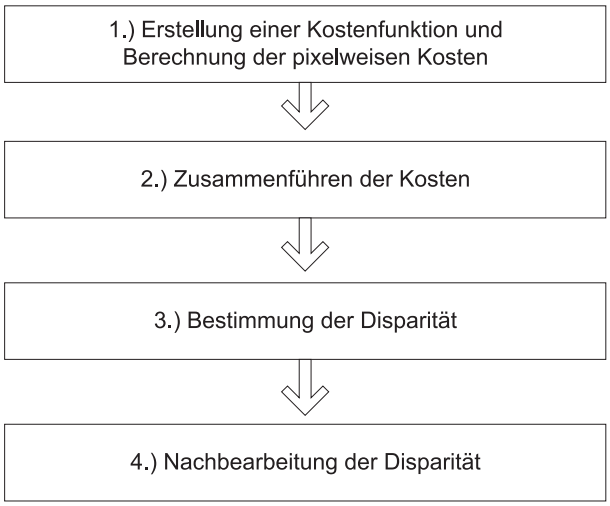


Abbildung 2.12.: Gliederung der Stereo-Matching-Verfahren [97]

Das am häufigsten verwendete Kostenmaß für die Ähnlichkeit zweier Grauwerte bildet die Summe der quadrierten Differenzen (SSD) bzw. die Summe der absoluten Differenzen (SAD). Letztere ist sehr einfach zu berechnen und wird in vielen echtzeitfähigen Stereo-Matchern verwendet.

$$\mathrm{SSD}(i,j,d) = \sum_{d} \left[I_{\mathrm{Basis}}(i,j) - I_{\mathrm{Match}}(i+d) \right]^2 \tag{2.26}$$

$$\mathrm{SAD}(i,j,d) = \sum_{d} \left| I_{\mathrm{Basis}}(i,j) - I_{\mathrm{Match}}(i+d) \right| \tag{2.27}$$

mit

- i, j = Koordinaten des Pixels im Bild
- d = Disparität
- I_{Basis} = Basisbild
- I_{Match} = Vergleichsbild

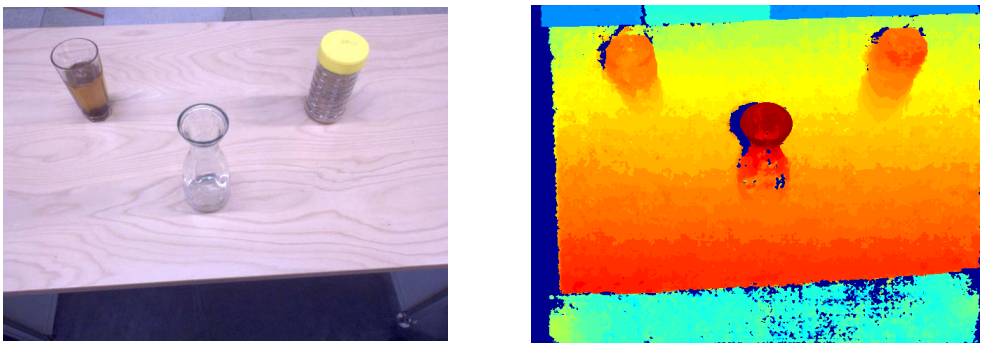
Aufgrund der relativ schlechten Matchingergebnisse, werden diese Kostenmaße typischerweise in Vorverarbeitungsschritten eingesetzt, in denen eine Start-Disparitätenkarte benötigt wird [64]. Weiterhin sind die SSD- und SAD-basierten Verfahren und deren Derivate wenig robust gegenüber Ausreißern. Dies gilt insbesondere in der zweiten Phase, in der die Kosten lokal aufsummiert werden. Limitierte (truncated) Kostenfunktionen grenzen die möglichen Kostenwerte ein. Die obere Schranke steht im Einklang mit dem Smoothness-Constraint. Dennoch sind SAD und SSD relativ empfindlich gegenüber Kontrast- und Helligkeitsunterschieden, die wiederum raumabhängig sind. Verursacht werden diese durch verschiedene Belichtungszeiten und Aufnahmevorrichtungen. Durch eine radiometrische Kalibrierung vor der Auf-

nahme kann nur ein Teil des Problems gelöst werden. Je nach dem Blickwinkel der Kamera auf das beleuchtete Objekt kann die am Detektor gemessene Lichtmenge unterschiedlich sein. Totalreflektionen wie in Abb. 2.13 sind ebenfalls schwer zuzuordnen. In Abb. 2.14 ist eine Szene im Original und mit einer Tiefenkarte dargestellt, die die

(a) Linkes Stereobild (b) Rechtes Stereobild

Abbildung 2.13.: Beispiel für Reflexion an transparenten Objekten

Schwierigkeiten bei der Korrespondenzanalyse an den betreffenden Stellen, die durch die Reflexionen und die Transparenzen verursacht werden, deutlich sichtbar macht. Diese Probleme sind in unkontrollierten Umgebungen nur sehr schwer zu beheben.



(a) Linkes Originalbild (H. Hirschmüller) (b) Tiefenbild

Abbildung 2.14.: Eine relativ schwierig zu analysierende Szene. Abbildung (b) wurde mit SGM, einem 9x7 Census und LR-Check sowie 3x3-Medianfilter erstellt.

Ein Offset bzw. Bias kann durch eine Mittelwertsubtraktion oder durch eine Umwandlung in ein Gradientenbild erster und auch zweiter Ordnung ausgeglichen werden.

Zero-Mean-Varianten des SAD und SSD sind der ZSAD und ZSSD

$$ZSAD(i,j,d) = \sum_d \left| [I_{\text{Basis}}(i,j) - \mu_{N(i,j)}] - [I_{\text{Match}}(i+d) - \mu_{N(i+d)}] \right| \tag{2.28}$$

$$ZSSD(i,j,d) = \sum_d \left| [I_{\text{Basis}}(i,j) - \mu_{N(i,j)}]^2 - [I_{\text{Match}}(i+d) - \mu_{N(i+d)}]^2 \right| \tag{2.29}$$

die eine geringe Verbesserung gegenüber den Standardvarianten liefern [46]. Die Grauwerte des Pixels p_i werden von dem Mittelwert $\mu_{N(p_i)}$ einer Nachbarschaft $N(p_i)$ befreit.

Census-Transformation

Sehr gute Eigenschaften bezüglich radiometrischer Unterschiede besitzt die Census-Transformation [55, 120]. Innerhalb eines Fensters werden die Grauwerte der Pixel $p_{k,l}$ aus der Nachbarschaft $k,l \in N_{i,j}$ mit dem zentralen Pixel $p_{i,j}$ verglichen.

$$C(p_{i,j}) = \begin{cases} 1 & \text{wenn } p_{k,l} < p_{i,j} \text{ mit } p_{k,l} \in N_{i,j}, \\ 0 & \text{sonst} \end{cases} \tag{2.30}$$

$$Census(i,j,d) = C(I_{base}(i,j)) \oplus C(I_{Match}(i+d,j)) \tag{2.31}$$

mit

- $N_{i,j}$ = Nachbarschaft der Pixel j und i
- $\oplus$ = Hammingdistanz (bitweiser XOR-Vergleich)
- $P_{k,l}$ = Pixel an der Position $k,l \in N_{i,j}$

Anschließend werden die durch Equation 2.30 erstellten Bitvektoren paarweise verglichen. Die Ähnlichkeit zwischen den beiden Bitvektoren dient als Kostenmaß für die Korrespondenzanalyse. Aufgrund des bitweisen Vergleiches zur Bildung der Hammingdistanz hat das Rauschverhältnis des Bildes einen starken Einfluss auf die Qualität des Kostenmaßes. Dieser Einfluss kann durch eine größere Fensterung insoweit kompensiert werden, als Census zumindest nicht schlechter als SAD bzw. SSD basierte Verfahren abschneidet [94].

Kostenaggregation

Für Echtzeitanwendungen haben sich lokale Matching-Methoden etabliert [54], [5], [67]. Zur Reduktion des Rauschens und zur Verringerung von Mehrdeutigkeiten werden die Matchingkosten von relativ kleinen Bildausschnitten von 3×3 bis 25×25 Pixel Größe zusammengefasst. Einige Kostenfunktionen wie zum Beispiel die auf Korrelation basierten Verfahren oder die Census-Transformation führen eine solche Fensterung schon von sich aus durch. Gleiches gilt für die globalen Ansätze, deren Kostenaggregation über die gesamte Bildgröße bzw. gesamte Zeile stattfindet.

Die zweite Phase aus Abb. 2.12 wird daher explizit für pixelweise Matchingkosten durchgeführt, um das SNR zu vergrößern. Daneben wird die Zuordnung homologer Punkte erleichtert, indem die Grauwerte eines Fensters zu mehr Variationsmöglichkeiten beitragen, die die Auflösung von Mehrdeutigkeiten vereinfacht.

Im zweidimensionalen Fall werden während der Zusammenführung der Kosten

$$C(i,j,d) = \sum_{k,l \in N_{i,j}} C(k,l,d)$$

mit

- $N_{i,j}$ = Nachbarschaft der Pixel j und i
- *C(k,l,d)* = Kostenfunktion mit den Argumenten k und l für die Pixelposition sowie der Disparität d

nur die Werte der Nachbarn in der Bildebene herangezogen, so dass frontal-parallele Oberflächen angenommen werden. Realistischer ist die Einbeziehung geneigter Oberflächen, wie zum Beispiel die eines Balles, so dass die Kostenaggregation nicht wie im 2D-Fall mit fester Disparität sondern in einem Disparitätsbereich

$$C(i,j,d) = \sum_{k,l,m \in N_{i,j,d}} C(k,l,m)$$

stattfindet.

Die eingangs erwähnte Verschmelzung an den Objektkanten kann reduziert werden, wenn die Größe und die Form der Fenster an die Objektkanten und die Disparitätsübergänge adaptiert wird. An dieser Stelle sind kleine Fenstergrößen wieder von Vorteil, so dass es immer ein Gegeneinanderwirken zwischen der Verringerung der Mehrdeutigkeiten und der Verwaschung von Objektkanten gibt. Ein sehr verbreiteter Fensteransatz vergleicht die Matchingkosten anhand einer festen Menge unterschiedlicher Fensterformen [28]. Als Kostenfunktion für die Fensterauswahl ist im einfachsten Fall die L1- oder L2-Norm nach Equation 2.26 bzw. Equation 2.27 verwendbar, wobei das Fenster mit den geringsten Kosten ausgewählt wird. In Abb. 2.15 sind für ein (3×3)-Fenster die möglichen Kombinationen dargestellt. Variable Fenstertechniken variieren solange die Größe des Fensters, bis ein Schwellwert bezüglich der Intensitätsverteilung erreicht worden ist [113]. Variable Fenster haben gute Eigenschaften in Bereichen mit wenig Texturierung, können jedoch nur schlecht auf Objektgrenzen reagieren.

Neben der Farbe [108], [15] und der Disparität [121] kann auch der optische Fluss zur Schätzung einer idealen Fensterform und -größe dienen [87]. Diese Verfahren sind jedoch aufwendiger in der Berechnung und bis zum heutigen Zeitpunkt nur bedingt echtzeitfähig. Besser geeignet für Real-Time-Anwendungen ist neben dem Algorithmus von O. Veksler [113] ein Ansatz von K. J. Yoon et al. [119], der dem bilateralen Filter von C. Tomasi et al. [106] in gewisser Weise sehr ähnlich ist. Anstelle fester

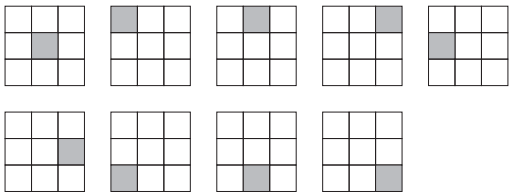


Abbildung 2.15.: Shiftable- oder auch Multi-Window genannte Verschiebung des zentralen Pixels

Fensterformen können durch eine individuelle Gewichtung der Fensterelemente anhand der Grauwertunterschiede und der räumlichen Nähe zum Fensterzentrum die Formen sehr variabel gestaltet werden.

Im Anschluss an die Phase 2, der Zusammenführung der Kosten, wird durch eine einfache Minimumsuche die Disparitätskarte erstellt und eventuell nachbearbeitet. Eine Übersicht der Summationsmethoden mit Hinblick auf Echtzeitanwendungen findet sich in M. Gong et al. [37].

Globale Matching Verfahren

Globale Verfahren hingegen lassen gewöhnlich den zweiten Schritt aus. Stattdessen wird die Bedingung der lokalen Glattheit mit Hilfe eines Terms $E_{\text{Glattheit}}(d)$ innerhalb einer globalen Energiefunktion

$$E(d) = E_{\text{Daten}}(d) + E_{\text{Glattheit}}(d) + E_{\text{Verdeckung}}(d) \tag{2.32}$$

modelliert. Die Equation 2.32 wird auf das vollständige Bild angewendet. Lokale Eigenschaften bzw. ein Kostenmaß für die Grauwerteunterschiede werden durch den Datenterm $E_{\text{Daten}}(d)$ eingebracht:

$$E_{\text{Daten}}(d) = \sum_{i,j} C(i, j, d(i, j)) \tag{2.33}$$

Durch Auswertung von Equation 2.33 spiegelt $E_{\text{Daten}}(d)$ die Kosten für die Parallaxe d wider, indem über das gesamte Bild die Kosten aus dem Kostenvolumen C aufsummiert werden.

Der Glattheitsterm $E_{\text{Glattheit}}(d)$ und der Verdeckungsterm $E_{\text{Verdeckung}}(d)$ geben die verschieden Einschränkungen des Suchraums wieder. Um die Komplexität von $E_{\text{Glattheit}}(d)$ im Rahmen zu halten, wird der Term nur für eine kleine lokale Umgebung definiert. In aller Regel handelt es sich dabei um die direkten Nachbarn an der Stelle (i, j). Die Aufstellung der Energiefunktion wird dadurch erleichtert, dass ein direkter Zusammenhang zwischen der Equation 2.32 und der bayesschen Statistik existiert [34]. Unter Verwendung von bayesschen Netzen und der Markov-Eigenschaft kann die Energiefunktion zudem als Graph dargestellt werden.

$$p(x|y) \propto p(y|x) \cdot p(x) \tag{2.34}$$

Die Werte y repräsentieren vorliegende Daten (Beobachtungen). Die Dichte $p(x)$ enthält Information über die Parameter x, bevor die Daten y erhoben wurden. Man nennt $p(x)$ daher die Prior-Verteilung für die Parameter x. In ihr ist die Vorinformation über die unbekannten Parameter zusammengefasst. Über die Dichte $p(y|x)$ gelangt die Information, die in den Daten y enthalten ist, zu den Parametern x. Da die Daten y vorliegen, wird diese Dichte nicht als Funktion der Daten y, sondern als Funktion der Parameter x interpretiert und als Likelihood bezeichnet. Die Dichte $p(x|y)$ wird als Posterior-Verteilung für die Parameter x bezeichnet. In Bezug auf das Korrespondenzproblem ist es möglich, $p(x|y)$ als eine diskrete Verteilung der Disparitäten in Abhängigkeit von der Bildinformation (Pixelposition) zu betrachten.

Das Korrespondenzproblem kann damit als Maximierungsaufgabe der Posterior-Verteilung (MAP) formuliert werden. Äquivalent zur Maximierung von $p(x|y)$ in Equation 2.34 ist die Darstellung in der Form einer Energiefunktion, die minimiert werden muss:

$$E(d) = E_d(d) + E_s(d) \tag{2.35}$$

Mit $E_d(d)$ wird der sog. Datenterm bezeichnet und die Likelihood modelliert. In $E_s(d)$ spiegelt sich das apriorische Wissen über die Disparitäten wider. Die Berechnung von einem d, das die Equation 2.44 minimiert, ist für den allgemeinen 2D-Fall NP-vollständig [12], so dass in der Regel eine Reihe von verschiedenen Approximationsverfahren zur Lösung verwendet werden. Dazu zählen Simulated-Annealing-Ansätze, die jedoch in der Praxis nicht geeignet sind. Belief-Propagation [25], [100], [73] Graph-Cuts [65] und die dynamische Programmierung [115], [114] sind effizientere Methoden und lassen sich ebenfalls unter bestimmten Voraussetzungen auf das Energie-Problem anwenden. Ein Vergleich zwischen Graph-Cuts und Belief-Propagation findet sich in M. F. Tappen et al. [104]. Eine allgemeine Betrachtung von Energiefunktionen, die als Bayes-Netze formuliert sind, liefert R. Szeliski et al. [102].

Zur besseren Darstellung werden die Randbedingungen mit Hilfe bedingter Wahrscheinlichkeiten graphisch anschaulich gemacht. Eine mögliche Formulierung des Korrespondenzproblems ist in Abb. 2.16 dargestellt. Für das zweidimensionale Bayes-Netz soll die Markov-Eigenschaft gelten. D.h. nur der direkt benachbarte Pixel hat Einfluss auf die Auswertung, so dass jedem Pixel $p_{i,j} = (x = i, y = j)^T$ eines Bildes ein Disparitätswert $d_{i,j}$ mit $d \in \{1 \dots n\}$ mit der bedingten Wahrscheinlichkeit $p(d_k|\tau_{i,j})$ zugewiesen wird.

Die runden Kästchen entsprechen Beobachtungen bzw. Daten $\tau_{i,j}$, die durch Kostenfunktionen ermittelt werden. Die eckigen Kästchen stellen die diskrete Verteilung der Disparitätszuweisung an der Stelle (i, j) dar.

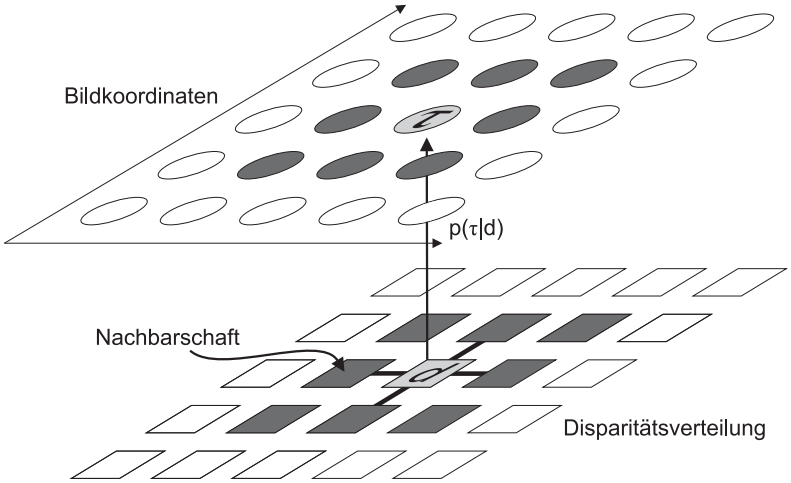


Abbildung 2.16.: 2D - Markov-Modell

Zwischen den Disparitätsstufen benachbarter Pixelpositionen existieren Abhängigkeiten. Diese örtlichen Randbedingungen werden durch die Kanten symbolisiert, die zwischen den eckigen Kästchen verlaufen. Mit Hilfe der Bayes-Regel kann die bedingte Verteilung $p(d_{(i,j)}|\tau_{(i,j)})$ von n möglichen Disparitäten aus dem Produkt des Prior-Wissens

$$p(d_{i,j}|N_{i,j})$$

und der Likelihood

$$p(\tau_{(i,j)}|d_{i,j})$$

approximiert werden. Abb. 2.16 zeigt ein Markov-Modell 1. Ordnung. Zur effizienten Berechnung des Prior-Wissens werden daher nur die Verteilungen der direkten Nachbarn $N_{i,j}$ (grau dargestellt) herangezogen. Die Lösung des Korrespondenzproblems entspricht einer Maximierungsaufgabe von Posterior-Verteilungen.

$$p(d|\tau) \propto \prod_{i,j} p(\tau_{i,j}|d_{i,j}) \cdot p(d_{i,j}|N_{i,j}) \tag{2.36}$$

mit

$$p(d_{i,j}|N_{i,j}) = \prod_{(k,l)\in N_{i,j}} G(d_{i,j}, d_{(k,l)}) \tag{2.37}$$

und

$$p(\tau_{i,j}|d_{i,j}) = \prod_{i,j} F(\tau_{i,j}, d_{i,j}) \tag{2.38}$$

mit

- $G(d_{i,j}, d_{k,l})$:= Wahrscheinlichkeitsverteilung des Disparitätswertes $d_{i,j}$ in Abhängigkeit der benachbarten Disparitätswerte
- $F(d_{i,j}, d_{k,l})$:= Wahrscheinlichkeitsverteilung des Disparitätswertes $d_{i,j}$ in Abhängigkeit zu den Beobachtungen

Äquivalent zur Maximierung von Equation 2.36 ist die Darstellung in der Form einer Energiefunktion E, die minimiert werden muss:

$$p(d|\tau) \propto \prod_{i,j} p(\tau_{i,j}|d_{i,j}) \cdot p(d_{i,j}|N_{i,j}) \tag{2.39}$$

$$= \prod_{i,j} F(\tau_{i,j}, d_{i,j}) \cdot \prod_{(k,l)\in N_{i,j}} G(d_{i,j}, d_{(k,l)}) \tag{2.40}$$

$$\propto \exp\left[-\sum_{i,j} F(\tau_{i,j}, d_{i,j}) - \sum_{i,j} \sum_{(k,l)\in N_{i,j}} G(d_{ij}, d_{(k,l)})\right] \tag{2.41}$$

$$E(d) := -log(p(d|\tau)) \tag{2.42}$$

$$= \sum_{i,j} F(\tau_{i,j}, d_{i,j}) + \sum_{i,j} \sum_{(k,l)\in N_{i,j}} G(d_{ij}, d_{(k,l)}) \tag{2.43}$$

$$= E_F(d) + E_G(d) \tag{2.44}$$

$E_F(d)$ ist der Datenterm aus Equation 2.33 und modelliert die Likelihood. In $E_G(d)$ spiegelt sich das Prior-Wissen über die Disparitäten wider und zum Beispiel der Local-Smoothness-Constraint.

Überdeckungen

Verdeckungen sind ein sehr häufig auftretendes Phänomen in der Stereobildverarbeitung, das auf unterschiedlichste Art und Weise modelliert werden kann. In Abb. 2.17 wird der Punkt P_1 nur von der linken Kamera erfasst, wodurch es nicht mehr möglich ist, dessen Tiefe zu triangulieren. Verdeckungen können prinzipiell auf drei Arten behandelt werden. Der einfachste und aus diesem Grund auch am häufigsten anzutreffende Ansatz basiert darauf, vor oder nach der Korrespondenzanalyse Überdeckungsbereiche herauszufiltern. Der Standardansatz dazu ist die Konsistenzüberprüfung (LR-Check, *left right check*). Je nach Schwellwert können sowohl mögliche Fehlzuweisungen als auch Überdeckungen festgestellt werden. Es werden zwei Disparitätskarten D_{BM} und D_{MB} berechnet, indem das Basis- und das Vergleichsbild miteinander vertauscht verarbeitet werden. BM steht für Basisbild (B) und zu matchendes Bild (M) und stellt den Ausgangszustand dar.

$$D(x) = \begin{cases} D_{\mathrm{BM}}(x), & \text{wenn } |D_{\mathrm{BM}}(x) - D_{\mathrm{MB}}(x - D_{\mathrm{BM}}(x))| \leq \varepsilon, \\ \text{ungültig}, & \text{sonst} \end{cases} \tag{2.45}$$

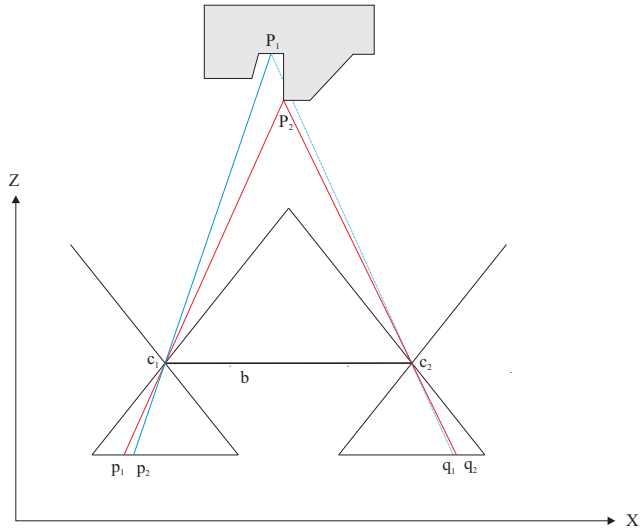


Abbildung 2.17.: Ein Beispiel für eine typische einseitige Verdeckung von Punkt P_1 [78]

Wird der Schwellwert ε überschritten, wie im rechten Fall aus Abb. 2.18, so gilt

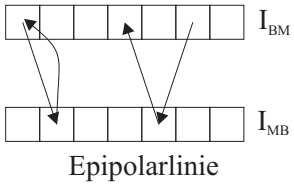


Abbildung 2.18.: Konsistenzüberprüfung

dieser Punkt als ungültig. Neben Verdeckungen werden auch Fehlzuweisungen erkannt, die durch perspektivische und radiometrische Verzeichnungen hervorgerufen werden (section 2.3). Durch geschickte Wiederverwendung von Zwischenergebnissen aus der Berechnung von D_{BM} kann die Verdopplung der Rechenzeit, während D_{MB} ermittelt wird, vermieden werden.

Das in section 2.4 erwähnte Ordnungsprinzip kann ebenfalls zur Erkennung von Überdeckungen genutzt werden. Eine Unterbrechung der monotonen Anordnung der homologen Punkte kann ein Indiz für eine Überdeckung sein. Häufig gilt auch, dass Überdeckungen mit Objektgrenzen durch starke Veränderungen der Grauwerte identifizierbar sind, so dass Kantenbilder als Vorwissen in die Erkennung von Überdeckung mit eingehen können.

Eine zweite Klasse von Verfahren zur Erkennung von Fehlzuweisungen beruht darauf, bereits in die Korrespondenzanalyse Algorithmen einzubinden, die gegenüber Verdeckungen robust sind. Die bereits erwähnten adaptiven Fensterfunktionen lokaler Matcher sind in diesem Kontext robuster als klassische quadratische Fensterfunktionen. Auch

hier kommt die Annahme zum Tragen, dass steile Grauwertgradienten auf Objektkanten und mögliche Überdeckungen schließen lassen.

Die letzte Klasse versucht, die Ursachen für Verdeckungen und Fehlzuweisung in gewisser Weise bereits bei deren Entstehung zu minimieren. Ein naheliegender Ansatz dazu ist es, die Objekte aus mehr als zwei Ansichten zu beobachten. Mit dem Ziel, jeden Punkt mindestens zweimal abzubilden, kann durch Multi-View-Methoden theoretisch jeder Objektpunkt trianguliert werden, mit dem vorteilhaften Nebeneffekt, dass der Fehler zwischen dem tatsächlichen Objektpunkt und dem triangulierten Pendant minimiert wird. In der Summation mehrerer SSD-korrelierter Bildpaare aus unterschiedlichen Perspektiven existiert bereits ein relativ einfaches Verfahren zur Fusion mehrerer Kameraaufnahmen [61].

Subpixelgenauigkeit

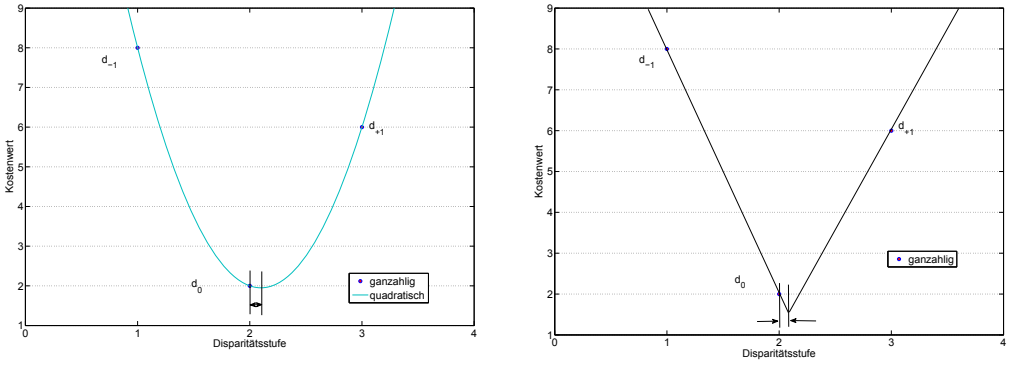
Für einige Applikationen kann es relevant sein, die Parallaxe mit Subpixelgenauigkeit anzugeben. Manche Stereo-Matching-Verfahren können bereits von sich aus die Disparität subpixelgenau berechnen [61]. Die überwiegende Mehrheit überlässt es jedoch einem Nachbearbeitungsschritt, die Disparität zu interpolieren. Eine übliche Methode beruht auf der Idee, drei Punkte mit einer Parabel zu interpolieren:

$$\Delta_d = \frac{d_{-1} - d_{+1}}{2 \cdot (d_{-1} - 2 \cdot d_0 - d_{+1})} \tag{2.46}$$

mit

- Δ_d = die Differenz zwischen d_0 und der Minimumstelle der Parabel
- $d_{-1/+1}$ = direkt benachbarten Disparitätswerte von d_0

Abb. 2.19a verdeutlicht diesen relativ einfachen Zusammenhang.



(a) Parabelfitting (b) Linearer Interpolationsansatz

Abbildung 2.19.: Interpolationsansätze zur Verfeinerung der Disparitätskarte

Die Gewinnung von Subpixelgenauigkeit durch Parabelfitting ist mit einem systematischen Fehler behaftet, der sich in einer starken Präferenz für ganzzahlige Werte niederschlägt. Eine Alternative besteht in der Verwendung einer stückweise linearen Funktion (Abb. 2.19b) anstelle der Parabel [99].

Verfeinerung der Disparitätskarte

Durch eine Reihe von Maßnahmen werden bereits berechnete Disparitäten im Nachhinein als ungültig markiert, da von ihnen angenommen wird, dass sie durch Fehler in der Aufnahme entstanden sind bzw. den Annahmen aus section 2.4 widersprechen. Dazu gehören:

- Durch Segmentierung der Eingangsbilder (Intensitäten) wird in erster Linie die Trennung zwischen texturierten Objekten und dem untexturierten Hintergrund verbessert.
- Es ist üblich, kleine Regionen der Disparitätskarte, die sich von ihrer direkten Nachbarschaft unterscheiden, als unplausibel anzusehen und für ungültig zu erklären. Durch Segmentierung der Disparitätskarte können diese Regionen identifiziert werden.
- Durch Medianfilter mit kleinen Fenstergrößen können einzelne Ausreißer eleminiert werden.
- Während der Konsistenzüberprüfung unterscheidet man zwischen verdeckten Bereichen und Fehlzuweisungen, wodurch letztlich Objektkanten und Hintergründe besser behandelt werden können.

Durch Anwendung der oben genannten Verfahren kann es dazu kommen, dass Teile der Disparitätskarte ungültig werden. Entstandene Lücken werden wiederum durch Interpolation nachträglich geschlossen. Dabei ist zu beachten, dass Fehlzuweisungen und überdeckte Bereiche gesondert behandelt werden müssen, um nachträgliche Verschmierungen der Disparitätsübergänge zu vermeiden. Die bereits erstellte Segmentierung kann für die Interpolation genutzt werden, um sicherzustellen, dass Vordergrundobjekte nicht in den Hintergrund hinein interpoliert werden. In H. Hirschmüller [52] wird ein effizientes Verfahren beschrieben, das Segmente auf Grundlage einer einfachen Nachbarschaftbeziehung erstellt. Innerhalb der Segmente werden die Werte bilinear interpoliert. Zwischen Segmenten wird extrapoliert, wobei die niedrigste Disparität gewählt wird.

Die Verfahren mit den besten Ergebnissen in Tabelle A.-4 wenden diese Methoden iterativ an, wodurch die Komplexität erheblich steigt.

3. Semi-Global-Matching-Algorithmus

Aufgrund seiner Eigenschaften ist der Semi-Global-Matching-Algorithmus (SGM-Algorithmus) für diese Arbeit von besonderer Bedeutung und soll im nachfolgenden Kapitel ausführlicher beschrieben werden.

3.1. Kostenaggregation

Der semi-globale Ansatz des gleichnamigen Algorithmus [50], [48], [51] geht von einer globalen Energiefunktion $E(d)$ aus, die über ein vollständiges Bild minimiert wird:

$$E(d) = \arg\min_{d} \left(E_d(d) + E_s(d)\right) \tag{3.1}$$

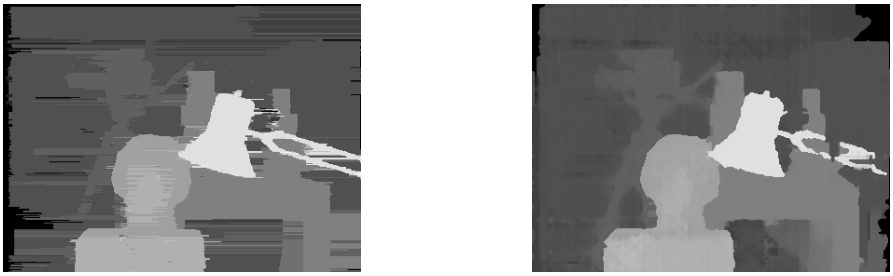
mit

- d = Disparität
- $E_d(d)$ = Datenterm in Abhängigkeit von d
- $E_s(d)$ = Smoothness- oder auch Glattheitsterm in Abhängigkeit der Disparität (vergleiche mit Equation 2.32)

Effiziente Approximationsverfahren, die das NP-vollständige Problem in polynomieller Zeit lösen, sind zum Beispiel Belief-Propagation [25], Graph-Cuts [65] und Dynamic-Programming [9], [8], [27] bzw. Scanline-Optimierung [97]. Die beiden letzteren Verfahren reduzieren die Berücksichtigung von $E_s(d)$ auf den eindimensionalen Fall. Ein typischer Effekt dieser Vereinfachung ist die horizontale Verschmierung entlang der horizontalen Abtastrichtung (Abb. 3.1a).

Das schlechte Ergebnis wird dadurch verursacht, dass die Randbedingungen nur in der horizontalen Richtung angewendet werden und keine bzw. nur eine sehr schwache Interaktion mit den vertikalen Bildinformationen stattfindet. Die Bedingungen, die in $E_s(d)$ formuliert sind, werden zu einseitig in waagerechter Richtung propagiert. Bemühungen, mehrere Scanzeilen parallel zu optimieren, lösen das Problem nur begrenzt.

Im Gegensatz zu dem unidirektionalen Ansatz des klassischen Dynamic-Programming bzw. der ähnlichen Scanline-Optimierung, approximiert der SGM-Algorithmus Equation 3.1 mehrdirektional durch Variationen der Scan-Reihenfolge (Abb. 3.2). Beide Ergebnisse sind in Abb. 3.1 anschaulich gegenübergestellt. Die Pfadkosten oder auch



(a) Dynamic-Programming [97] (b) SGM

Abbildung 3.1.: Vergleich zweier Matching-Ergebnisse

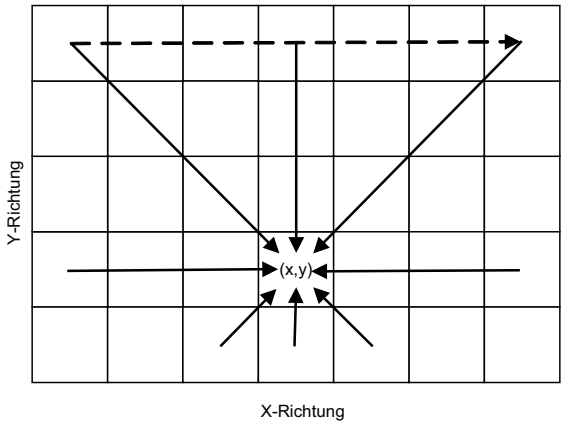


Abbildung 3.2.: Acht gleichverteilte 1D-Optimierungspfade propagieren die Constraints aus Equation 3.1approximativ in die Ebene. Die typische physikalische Abtastfolge ist gestrichelt dargestellt.

Richtungskosten $L_r(i,j,d)$ werden nach wie vor mit Hilfe der dynamische Programierung für jede Scan-Reihenfolge ermittelt, jedoch in einem anschließenden Summationsschritt

$$S(i,j,d) = \sum_r L_r(i,j,.) \tag{3.2}$$

mit den Zwischenergebnissen der weiteren Richtungen zusammengefasst. Die Kosten $L_r(i,j,d)$ eines Pfades r sind rekursiv definiert.

$$L_r(i,j,d) = C(i,j,d) + min(A,B,C,D) - \min_k(L_r(i-r,j-r,k)) \tag{3.3}$$

mit

- $C(i,j,d)$ = pixelweise Matchingkosten
- $L_r(i,j,.)$ = Kostenvektor (Pfadkosten) an der Stelle (i,j)
- r = aktuelle Richtung

- (i, j) = Stelle (i,j)
- $(i - r, j - r)$ = Vorgängerstelle von (i,j) der entsprechenden Richtung r
- A = $L_r(i - r, j - r, d)$
- B = $L_r(i - r, j - r, d - 1) + P1$
- C = $L_r(i - r, j - r, d + 1) + P1$
- D = $\min_k(L_r(i - r, j - r, k)) + P2$
- P1 = Strafkosten für kleine Disparitätssprünge
- P2 = Strafkosten für große Disparitätssprünge

3.2. Kostenfunktion

$C(i, j, d)$ sind die pixelweisen Matchingkosten, die durch eine beliebige Kostenfunktion erzeugt werden können. In der ursprünglichen Veröffentlichung von H. Hirschmüller [51] werden pixelweise Kosten auf der Basis eines SAD-Derivats und alternativ dazu durch Mutual-Information gebildet. Eine Implementierung von S. K. Gehrig et al. [31] verwendet zum Beispiel das ZSAD-Kostenmaß. Die Strafkosten $P1$ und $P2$ sind notwendig, um zu geringe Kostenwerte $C(i, j, d)$ zu kompensieren, die durch Fehlzuweisungen entstehen. $P1$ sollte kleiner gewählt werden als $P2$, da kleine Disparitätssprünge gegenüber großen bevorzugt werden. $P2$ ist zusätzlich sensitiv auf Grauwertänderungen eingestellt.

$$P2 = \frac{P2'}{|p_{(i,j)} - p_{(i+d,j)}|} \tag{3.4}$$

Starke Unterschiede in den Grauwerten fallen häufig mit Objektgrenzen zusammen, so dass eine dynamische Gestaltung von $P2$ sinnvoll ist.

Equation 3.3 wird rekursiv berechnet und unterscheidet sich von der klassischen dynamischen Programmierung dadurch, dass das Ordnungsprinzip nicht berücksichtigt werden muss, Überdeckungen werden ebenfalls nicht explizit betrachtet. Die Berechnungsvorschriften aus den Gleichungen 3.3 und 3.2 entsprechen dem Scanline-Verfahren aus [97]. In gewisser Art und Weise stellen sie eine auf die eindimensionale Komponente reduzierte Form von Equation 3.1 dar.

Durch Subtraktion von $\min_k(L_r(i-r, j-r, k))$ kann eine obere Schranke von $L_r(i, j, d)$ ermittelt werden, wodurch die notwendige Anzahl an Bits zur Darstellung von $L_r(i, j, d)$ eingeschränkt wird:

$$L_r(i, j, d) \leq C_{\max} + P2 \tag{3.5}$$

In Anlehnung an Equation 3.5 kann auch für $S(i, j, d)$ ein maximaler Wert bestimmt

werden und hängt von der Anzahl der durchlaufenen Pfade r ab.

$$S(i,j,d) \leq |r| \cdot (C_{\max} + P2) \tag{3.6}$$

In Abhängigkeit von der gewählten Anzahl der Richtungen und der physikalischen Abtastfolge des Bildsensors können nicht alle Richtungen gleichzeitig berechnet werden, so dass ein Teil von S zwischengespeichert werden muss.

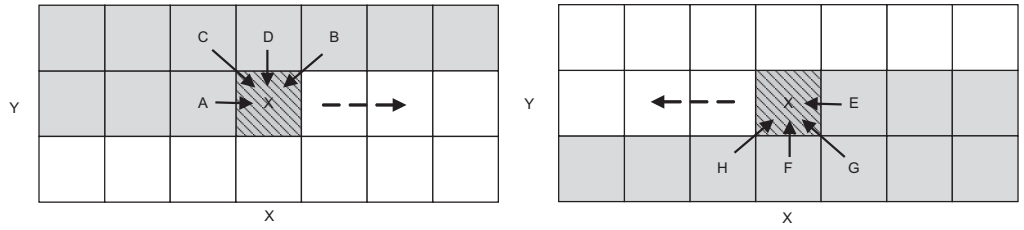


Abbildung 3.3.: In zwei Phasen werden die unterschiedlichen Pfadrichtungen durchlaufen.

Eine natürliche Vorgehensweise, die einzelnen Pfade effizient zu berechnen, besteht darin, erst die oberen Richtungen zu bestimmen und nach Equation 3.2 zusammen zufassen. Abb. 3.3 verdeutlicht, wie in einer zweiten Phase die restlichen unteren Pfade berechnet werden.

3.3. Berechnung der Disparität

Erst wenn die Pfadkosten für alle Richtungen eines Pixels ermittelt worden sind, werden die minimalen Kosten $D(i,j)$ bestimmt:

$$D(i,j) = \arg\min_{d}(S(i,j,d)) \tag{3.7}$$

Diese Herangehensweise entspricht der WTA-Strategie (*winner takes all*). Für eine Sub-Pixel-Verfeinerung kann ein Polynom zweiter Ordnung an der Stelle $d(i,j) = \arg\min_d S(i,j,d)$ und den benachbarten Disparitäten zur Interpolation entsprechend Abb. 2.4 verwendet werden:

$$\Delta_d(i,j) = \frac{S_{-1} - S_1}{2 \cdot (S_{-1} - 2 \cdot S_0 - S_1)} \tag{3.8}$$

mit

- $S_1 = S(i,j,d(i,j)+1)$
- $S_0 = S(i,j,d(i,j))$
- $S_{-1} = S(i,j,d(i,j)-1)$

Die gewählte Parabel ist eine ausreichend gute Approximation für beliebige Kostenfunktionen [99]. Das Ergebnis ist eine sub-pixel genaue Tiefenkarte D_{sub}:

$$D_{sub}(i,j) = D(i,j) + \Delta_d(i,j) \tag{3.9}$$

3.4. Verfeinerungsschritte

Falsch zugeordnete Disparitäten, die auf Teilüberdeckungen zurückgeführt werden können, können durch eine L/R-Konsistenzüberprüfung (Abb. 2.4) erkannt werden. Dazu sind die Disparitätskarten D_{BM} und D_{MB} notwendig. D_{MB} kann durch vollständige Neuberechnung oder durch eine sog. schräge Suche, wie in Equation 3.10 definiert, berechnet werden. Letztere Variante hat den Vorteil, die Laufzeit und den Speicherbedarf zu minimieren. Bessere Ergebnisse werden jedoch mit einer vollständigen Neuberechnung erzielt:

$$D_{\mathrm{BM}}(i,j) = \arg\min_{d}(S(\mathrm{epipolar}_{\mathrm{M}}(i,j),d)) \tag{3.10}$$

mit

- D_{BM} = Disparitätskarte, die durch die Anordnung *Basisbild - Vergleichsbild* erstellt wurde
- D_{MB} = Disparitätskarte, die durch die Anordnung *Vergleichsbild - Basisbild* erstellt wurde
- $\mathrm{epipolar}_{\mathrm{M}}(i,j)$ = Pixelposition entlang der Epipolarlinie im Vergleichsbild

Der Punkt $D(i,j)$ wird als ungültig markiert, wenn gilt[1]:

$$D(i,j) = \begin{cases} D_{\mathrm{BM}}(i,j), & \text{wenn } |D_{\mathrm{BM}}(i,j) - D_{\mathrm{MB}}(i - D_{\mathrm{BM}}(i,j),j)| \leq 1, \\ \text{ungültig}, & \text{sonst} \end{cases} \tag{3.11}$$

Disparitätswerte, die stark von denen ihrer Nachbarschaft abweichen und als Ausreißer bzw. als ungültig angenommen werden können, lassen sich zum Beispiel durch einen Medianfilter oder bilateralen Filter unterdrücken. Weiterhin können kleine Segmente mit nur wenigen Pixeln und identischen Disparitätswerten als ungültig markiert werden, da diese häufig Fehlzuweisungen sind. Zur Verbesserung der Oberflächenmodellierung kann es nützlich sein, Lücken in der Disparitätskarte zu interpolieren. Lücken werden zum Beispiel durch die Anwendung der L/R-Konsistenzüberprüfung verursacht, die entweder auf Überdeckungen oder Fehlzuweisungen zurückzuführen ist. Damit die relativ scharfen Objektgrenzen in der Disparitätskarte nicht durch die Interpolation nachträglich verschmiert werden, muss zwischen verschiedenen Situationen der Szene unterschieden werden. Gelöschte Disparitätswerte, die durch Fehlzuweisungen entstanden sind, können von jedem benachbarten Pixel aus interpoliert werden, solange diese sich nicht an Objektgrenzen befinden. Ungültige Disparitäten, die durch

[1] Es wird angenommen, dass die Epipolarlinie parallel zur Abtastfolge in i-Richtung verläuft.

Überdeckungen hervorgerufen werden, dürfen hingegen nur aus Hintergrundbereichen extrapoliert werden. Abb. 3.4 veranschaulicht ein Verfahren, anhand dessen ungültige

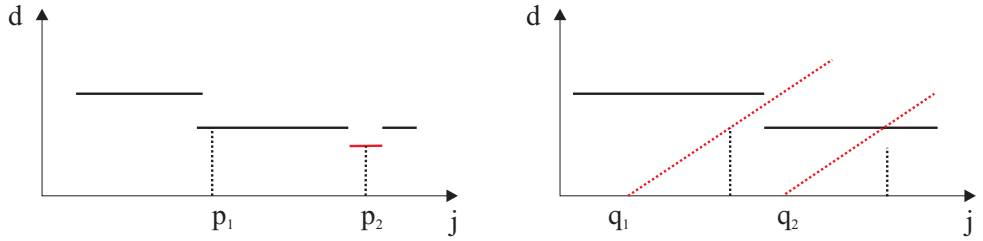


Abbildung 3.4.: Horizontaler Schnitt durch die Disparitätskarten D_{BM} (links) und D_{MB} (rechts) als Stufenfunktion dargestellt

Disparitäten, verursacht durch Fehlzuweisungen und Überdeckungen, unterschieden werden können. Eine der roten (diagonal verlaufende, gestrichelte) Linien im rechten Bild ist die Epipolarlinie $\mathrm{epipolar}_{\mathrm{MB}}(q_1, j, d)$ und $\mathrm{epipolar}_{\mathrm{MB}}(q_2, j, d)$ die zweite. Der falsche Disparitätswert $D(p_1, j)$ ist durch eine Überdeckung begründet und dadurch zu erkennen, dass die Epipolarlinie $\mathrm{epipolar}_{\mathrm{MB}}(q_1, j, d)$ sich mit keinem Punkt aus der als Ebene gedachten Disparitätskarte D_{MB} schneidet. Die Gerade $\mathrm{epipolar}_{\mathrm{MB}}(q_2, j, d)$ hingegen schneidet die Stufenfunktion D_{BM} an der Stelle $D(p_2, j)$ und deutet daher auf eine Fehlzuweisung hin.

3.5. Kachelung

Aufnahmen, die mit hochauflösenden Kameras wie der UltraCam[2], der MFC[3] oder der ADS40[4] etc. gemacht wurden, sind zu groß, als dass der SGM-Algorithmus und Standard-Hardware sie im Ganzen verarbeiten können. Das SGM-Verfahren benötigt zu viel Zwischenspeicher sowohl für die Richtungen als auch zum temporären Speichern des Volumens S. Eine Möglichkeit, die Luftbildaufnahme mit dem SGM dennoch zu berechnen, besteht darin, die einzelnen Bilder in Kacheln zu zerlegen. Die Kacheln werden mit einer geringen Überlappung erstellt, um die nur unvollständig vorhandene Information an den Bildrändern zu kompensieren. Benachbarte Kacheln werden mittels einer Gewichtsfunktion, die am Rand linear abfällt, verschmolzen. Abb. 3.5 veranschaulicht die Überlappung der Kacheln. Disparitätszuweisungen an den Kachelrändern bekommen eine geringe Gewichtung, da die Pfadinformation für mindestens drei Richtungen unvollständig ist bzw. gänzlich fehlt. Die Kachelgröße kann beliebig gewählt werden, so dass die zur Verfügung stehenden Ressourcen maximal genutzt werden. Für den Fall, dass eingangs die Luft- oder Satellitenaufnahmen bereits zu groß sind, um vollständig in den RAM geladen zu werden, können diese nach dem gleichen Muster in Subbilder bzw. Makrokacheln zerlegt werden. Eine Hierarchie von

[2] www.microsoft.com/en-us/ultracam/
[3] elib.dlr.de/53227/
[4] elib.dlr.de/46634/

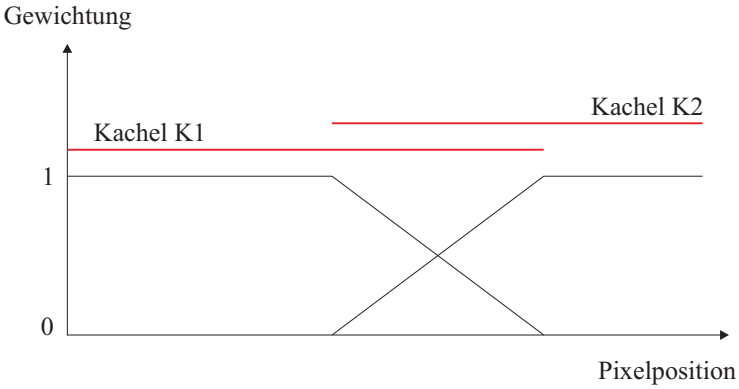


Abbildung 3.5.: Überlappung der Kacheln und Gewichtung der Kachelränder [51]

Kacheln entsteht, die die Hardware optimal ausnutzt (Abb. 3.6). Die Fusionierung

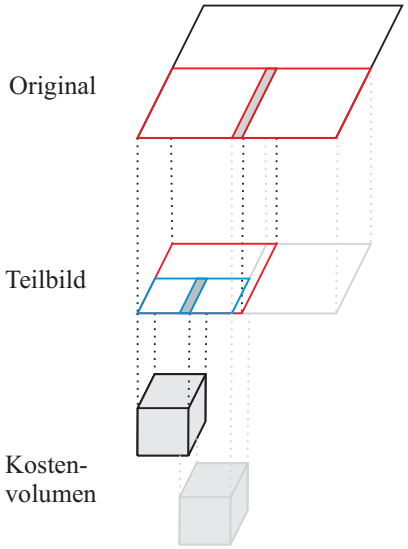


Abbildung 3.6.: Hierarchie der Kacheln und Überlappung der Kachelränder

von Mehrfachaufnahmen und mögliche Nachbearbeitungsschritte wie zum Beispiel der L/R-Check zum Erkennen von Überdeckungen und Falschzuweisungen werden für jede Makrokachel separat durchgeführt. Die Disparitätskarten werden ebenfalls mit einer Gewichtung an den Rändern verschmolzen. Mit Hilfe der Kachelung der Sensordaten ist es möglich, beliebig große Datensätze zu bearbeiten, so dass an dieser Stelle keine Einschränkungen gemacht werden müssen. Daneben ist die Kachelung von erheblicher Bedeutung für die Verteilung der Bilddaten auf verschiedene Rechenknoten. So können mehrere unabhängige Instanzen des SGM-Algorithmus parallel auf die Eingangsbilder angesetzt werden und daraus ein Zeitvorteil bei der Bearbeitung gewonnen werden.

3.6. Komplexität

Unabhängig von der gewählten Kostenfunktion müssen alle Pixel 8 bzw. 16mal besucht werden, um Equation 3.2 zu berechnen. Die Ausführung von Equation 3.3 setzt voraus, dass alle Disparitätsstufen von $d \in \{1, \dots d_{\max}\}$ betrachtet werden. In Abhängigkeit zu der Ausführungsgeschwindigkeit ergibt sich somit eine Komplexität des Kernalgorithmus von

$$\mathcal{O}(W \cdot H \cdot d_{\max} \cdot \text{Bildfrequenz}) \subseteq \mathcal{O}(n^4)$$

mit

- W = Bildbreite
- H = Bildhöhe
- $d_{\max}$ = maximale Anzahl an Disparitätsstufen

Das Pfadkostenvolumen S wird in zwei Phasen berechnet. In Phase Eins werden die Richtungen A,B,C und D parallel ermittelt, wodurch

$$L \in \mathcal{O}((W \cdot d_{\max} \cdot 3 + d_{\max}) \cdot 2 \text{ Bytes}) \subseteq \mathcal{O}(n^2 + n)$$

interner Speicher für vier Richtungsvektoren $L_A \dots L_B$ notwendig ist. Die Richtungen B,C und D müssen für eine gesamte Zeilenlänge gepuffert werden, wohingegen A aus dem unmittelbaren Vorgänger in Abtastreihenfolge herrührt. Das gleiche gilt noch einmal für die Richtungen E, F, G und H aus der zweiten Phase.

Aufgrund der Aufteilung des Algorithmus in zwei Phasen müssen die Ergebnisse aus der ersten Summation der Equation 3.2 ebenfalls zwischengespeichert werden. Zur Speicherung der aufsummierten Pfadkosten S sind

$$S \in \mathcal{O}(W \cdot H \cdot d_{\max} \cdot 2 \text{ Bytes}) \subseteq \mathcal{O}(n^3)$$

Bytes notwendig. Für ein VGA-Bild mit einer gewünschten Tiefenauflösung von 64 Pixeln ergibt sich somit ein Speicherbedarf von $37,5$ MB für S und ca. 241 KB für die temporäre Speicherung der Richtungen. Abbildung 3.7 stellt schematisch den Datenfluss dar.

Erheblich mehr Speicherbedarf benötigt das SGM-Verfahren für die Berechnung von Luftbildaufnahmen. Die UltraCamLp von Microsoft/Vexcel zum Beispiel liefert Bilder mit einer Auflösung von ca. 11500·7500 Pixel. Aufgrund des sehr hohen Speicherbedarfs sowohl von ca. 160 GB für S als auch für die Speicherung der Richtungen von ca. 66 MB können diese Datensätze nicht ohne weiteres in den RAM geladen werden.

Ähnlich hohe Werte für den Speicherbedarf werden durch die Multi-Functional-Camera (MFC) [11] generiert. Die MFC ist eine digitale Zeilenkamera, die vom DLR für die Luftbildauswertung entwickelt wurde. Sie wird zur Erzeugung hochwertiger 3D-

Kamerasystem	räumliche Auflösung	max. Disparität	Speicherbedarf für L	Speicherbedarf für S
VGA	640×480	64	241 KB	37,5 MB
VGA	640×480	128	480 KB	75 MB
Gen	1024×1024	64	384 KB	128 MB
Gen	1024×1024	128	769 KB	256 MB
UltraCam	11500×7500	500	33 MB	80 GB
UltraCam	11500×7500	1000	66 MB	160 GB

Tabelle 3.1.: Beispiele für den zu erwartenden Speicherbedarf für SGM

Gelände- und Oberflächenmodelle verwendet. Es existieren mehrere Varianten der MFC, wovon die letzte mit einer 10k-Zeile ausgestattet ist.

Der Speicherbedarf ist in der Table 3.1 für die VGA, UltraCam und einer exemplarischen $1024 \cdot 1024$-Industriematrixkamera (Gen) zusammengefasst.

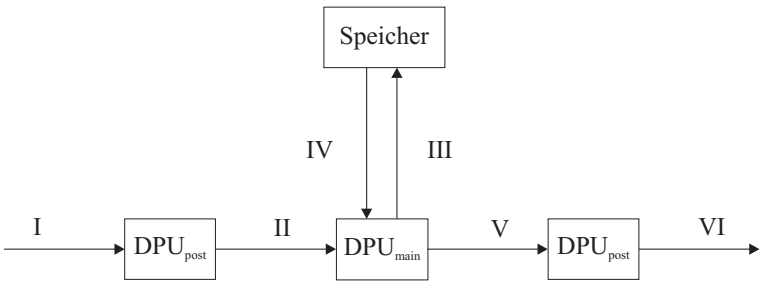


Abbildung 3.7.: Datenfluss-, DPU- und Speicherübersicht

$$
\begin{aligned}
\text{I} \leftarrow &\quad \text{Höhe} \cdot \text{ Breite} \cdot 2 \text{ Phasen} \cdot \text{Bildfrequenz} \cdot 1 \text{ Byte} \\
\text{II} \leftarrow &\quad \text{Höhe} \cdot \text{ Breite} \cdot 2 \text{ Phasen} \cdot \text{Bildfrequenz} \cdot 1 \text{ Byte} \\
\text{III} \leftarrow &\quad \text{Höhe} \cdot \text{ Breite} \cdot 2 \text{ Phasen} \cdot \text{Bildfrequenz} \cdot \text{Disparität} \cdot 2 \text{ Byte} \\
\text{IV} \leftarrow &\quad \text{Höhe} \cdot \text{ Breite} \cdot 2 \text{ Phasen} \cdot \text{Bildfrequenz} \cdot \text{Disparität} \cdot 2 \text{ Byte} \\
\text{V} \leftarrow &\quad \text{Höhe} \cdot \text{ Breite} \cdot 2 \text{ Phasen} \cdot \text{Bildfrequenz} \cdot 1 \text{ Byte} \\
\text{VI} \leftarrow &\quad \text{Höhe} \cdot \text{ Breite} \cdot 2 \text{ Phasen} \cdot \text{Bildfrequenz} \cdot 1 \text{ Byte}
\end{aligned}
$$

Die beiden oben gezeigten Beispiele verdeutlichen, dass an der Schnittstelle III und IV aus Abb. 3.7 ein möglicher Engpass bezüglich der Datentransferrate zu erwarten ist. Die Eingangs- und Ausgangsdatenraten sind jedoch moderat und können aus heutiger Sicht mit Standardkomponenten zur Datenübertragung bewältigt werden.

4. Plattformen

Im folgenden Kapitel werden die grundlegenden technischen Aspekte dieser Arbeit behandelt, indem die allgemeinen Prinzipien und Hardwarearchitekturen einleitend vorgestellt werden. Einen Haupteil dieses Kapitels nimmt die konfigurierbare Hardware und eine dazu in Bezug stehende Entwurfsmethodik ein.

4.1. Hardwarearchitekturen

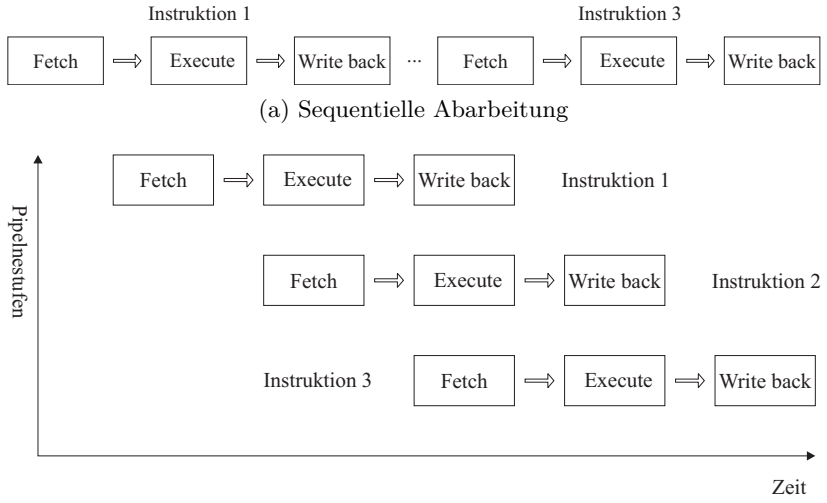
Traditionell wurden Applikationen in Software geschrieben und diese sequentiell von Mehrzweck-Prozessoren (CPUs) bearbeitet. Ein klassischer Compiler übersetzt das von Menschenhand geschriebene Programm in eine endliche Menge von Instruktionen, die einzeln hintereinander ausgeführt werden und den Prozessor für diese Zeit blockieren. Die gesamte Berechnungsdauer für die Lösung der Rechenaufgabe entspricht genau der Summe, die die einzelnen Instruktionen benötigen. Die Zeit, in der sich der Prozessor im Leerlauf befindet, spielt keine Rolle.

Eine Verringerung der Gesamtberechnungsdauer ist nur dann möglich, wenn zum einen die einzelnen Befehle zum Beispiel durch höhere Taktraten der CPU schneller ausgeführt oder zum anderen, wenn die Instruktionen zeitlich parallel ausgeführt werden. Klassische x86-CPUs können heute dank moderner Fertigungsprozesse Taktraten im Gigahertzbereich erreichen und sind damit um ein Vielfaches leistungsfähiger als noch vor 10 Jahren. Grundsätzlich ist jedoch die Taktrate begrenzt durch die thermische Verlustleistung, die durch das schnelle Umladen der Transistoren entsteht, und durch die elektrischen Eigenschaften des Substrats.

Neben der Zeit für die Berechnung der einzelnen Instruktionen wird die Gesamtlaufzeit zusätzlich von der Leistungsfähigkeit der Schnittstellen und des peripheren Speichers im und um den Prozessor herum bestimmt. Instruction-Level-Pipelining kann dazu verwendet werden, den Leerlauf des Prozessors, der zum Beispiel durch Lese- und Schreibzugriffe verursacht wird, zu minimieren. Instruktionen werden in kleinere Einheiten zerlegt, die sobald wie möglich ausgeführt werden, ohne auf die Beendigung einer vorhergehenden Instruktion zu warten. Die sequentielle Ausführung der Instruktionen eins bis drei in Abbildung 4.1a erfordert exemplarisch neun Zyklen, wohingegen das Pipelining der Teilinstruktionen in drei Stufen zu einer Verkürzung auf fünf Zyklen führt (Abb. 4.1b).

Häufig vorkommende Begriffe im Zusammenhang mit paralleler Abarbeitung sind:

- Pipelining: Verkleinerung der Taktzeit durch Aufbrechen der Instruktionen



(a) Sequentielle Abarbeitung

(b) Pipeline

Abbildung 4.1.: Instruction-Level-Pipelining

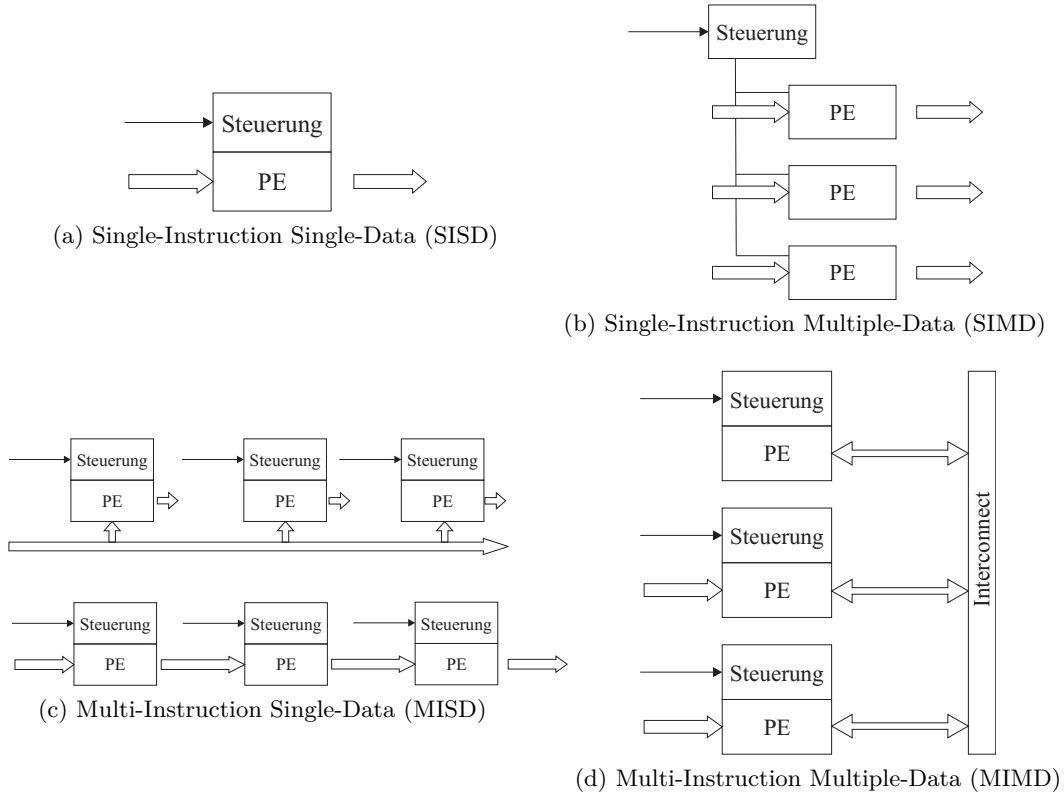
- Superskalarität: parallele Verwendung multipler Verarbeitungseinheiten
- Multiprozessoren: mehrere Rechnerkerne
- Vektorprozessoren (SIMD, Abb. 4.2), Datenflussarchitekturen
- analog auf Anwendungsebene: Multitasking, Threading usw.

Beim Instruction-Level-Pipelining wird durch zusätzliche Logik in der Hardware der Leerlauf des Prozessors minimiert. Der höchste Optimierungsgrad ist mit einer vollständig gefüllten Pipeline erreicht. Verzweigungen und rekursive Abhängigkeiten erzwingen jedoch im Regelfall eine unvollständig gefüllte Pipeline, die immer wieder geleert werden muss. Im schlimmsten Fall nimmt die Abarbeitung der Instruktionen die klassische Form an und die zusätzlich eingeführte Logik für die Pipeline bleibt ungenutzt. Dennoch besitzen heutige Prozessoren um die 10 bis 30 Pipelinestufen.

Ein ähnliches Konzept zur Steigerung der Auslastung eines Prozessors wurde mit der Entwicklung superskalarer Prozessoren umgesetzt, die in der Lage sind, mehrere Instruktionen parallel auf redundant vorhandenen Teilen des Prozessors zu bearbeiten (auch grobe Instruction-Level-Parallelität genannt). Zusätzliche Logik im Prozessor entscheidet zur Laufzeit (superskalar) oder der Compiler durch spezielle Befehle (Very-Large Instruction Words), welche Instruktionen parallel ausführbar sind. Eine Erweiterung dieser Hardwareredundanz auf mehrere Prozessoren spiegelt sich im Multi-Core- bzw. im Multi-Prozessoransatz wider, wobei an dieser Stelle auf Thread-Ebene die Applikation zeitlich parallel gelöst wird (Thread-Level-Parallelität). Heutige High-Performance-Prozessoren sind zu einem hohen Grad hybride Systeme, die die Konzepte des Instruction-Pipelining in superskalaren Multi-Prozessor-Systemen vereinen.

Analog zum Pipelining von Instruktionen ist es sinnvoll, den Datentransfer in glei-

cher Art und Weise zu staffeln. Vektormaschinen sind im Gegensatz zum klassischen Ansatz in der Lage, mit einer Instruktion mehrere Datenwörter zu manipulieren. Die Zeitdauer, die bis zum Transfer eines Datenwortes aus dem Speicher vergeht, kann somit entscheidend gesenkt werden. Eine gängige Einteilung paralleler Architekturen [26], die im weiteren Verlauf verwendet wird, basiert darauf, relativ einfach Instruktionen und Daten zueinander in Beziehung zu setzen. Abb. 4.2 stellt eine Übersicht über die Zusammenhänge von Instruktionen und Datenfluss dar. Single-Instruction Single-Data sind die bereits erwähnten traditionellen Einprozessor-Architekturen.



(a) Single-Instruction Single-Data (SISD)

(b) Single-Instruction Multiple-Data (SIMD)

(c) Multi-Instruction Single-Data (MISD)

(d) Multi-Instruction Multiple-Data (MIMD)

Abbildung 4.2.: Zusammenhang zwischen Daten und Instruktionen [26]

Eine Vektormaschine gehört hingegen zur Kategorie der Single-Instruction-Multiple-Data-Architekturen (SIMD). Diese Fähigkeit der Vektorverarbeitung war ursprünglich nur in Supercomputern wie der Cray-X1 zu finden. In heutigen x86-Prozessoren sind SIMD-Operationen ebenfalls möglich und finden sich in den Befehlssatzerweiterungen SSE, AltiVec und 3DNow! etc. wieder. Typische Operationen in der Bildverarbeitung wie zum Beispiel die Anwendung eines lokalen Fensters können mit diesen Befehlserweiterungen gegenüber nicht SIMD-Prozessoren beschleunigt werden. Dennoch ist die klassische CPU besser für instruktions-intensive Anwendungen geeignet.

Digitale Signalprozessoren (DSPs) sind Mikroprozessoren mit einem relativ kleinen

Befehlssatz, die speziell für komplexe Rechenoperationen wie zum Beispiel Fest- bzw. Gleitkommaarithmetik entwickelt worden sind, die in der Signalverarbeitung typisch sind. Filterfunktionen und Transformationen lassen sich mit DSPs effizienter realisieren als mit CPUs. Die Gründe dafür sind vielseitig. Zum einen werden die Konzepte zur Hardwarebeschleunigung von der CPU übernommen (superskalar, SIMD) und zum anderen die Arithmetikeinheiten auf die Funktionen der Signalverarbeitung angepasst (FusedMAC etc.). Spezielle Memory-Management-Einheiten (MMUs), Caching-Strategien und DMA-Transfers optimieren den Datentransfer ebenfalls anwendungsspezifisch, so dass DSP sehr kundenspezifisch sind. Aufgrund ihres geringen Energieverbrauchs und ihrer kompakten Größe haben sie ihren Platz vor allem in eingebetteten Systemen. Die Grenze zwischen Standardprozessoren und DSPs verläuft fließend, da mittlerweile Prozessoren um DSP-Elemente (ARM Cortex A9 + DSP, TI OMAP-Serie) erweitert werden. Eine weitere Gruppe unter den Mikroprozessoren

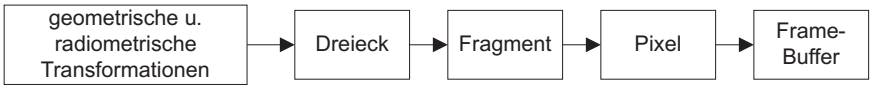


Abbildung 4.3.: Traditionelle GPU-Pipeline [90]

sind die Graphic-Processing-Units (GPUs), die insbesondere in den letzten Jahren an Bedeutung als Hardwarebeschleunigung für eine allgemeine Klasse von Rechenaufgaben gewonnen haben. Die Hauptaufgabe der GPUs ist das Rendern von 2D- und 3D-Vektormodellen, wofür die GPU über eine Vielzahl paralleler Fließkomma-Recheneinheiten verfügt. In ähnlicher Art und Weise wie die DSPs, sind die GPUs ursprünglich für diesen sehr speziellen Aufgabenbereich entwickelt worden. Die GPU verfügt aus diesem Grund über einen sehr kleinen Befehlssatz, wie er zur effizienten Berechnung von Pixeln nötig ist. Die Hardware ist ebenfalls auf den massiv parallelen Datenstrom von geometrischen Beschreibungen und Grauwerten zugeschnitten. Die klassische GPU-Hardware ist eine typische Pipeline-Architektur mit auf die Teilaufgaben zugeschnittenen Hardwarestufen (Abb. 4.3). Am Anfang der Bearbeitung erhält die GPU Befehle und Vertex-Daten von der CPU, die in einem ersten Schritt im Vertex-Shader geometrischen und radiometrischen Transformationen ausgesetzt werden. Anschließend wird die Information in Dreiecke, Fragmente und Pixel umgerechnet und im Frame-Buffer zu Ausgabe am Monitor abgelegt. Die bereits vorgestellten Konzepte der Daten- und Befehlsparallelität lassen sich auch in GPUs wiederfinden. Heutige GPUs sind zum Teil superskalar, wodurch mehrere Vektor- und Matrixoperationen parallel ausgeführt werden können. GPUs sind ebenfalls SIMD-orientiert, jedoch unterscheiden sich zum Beispiel GPUs und DSPs an einigen Stellen. Die Taktraten einer GPU bewegt sich im Gigahertzbereich, während DSPs diese nicht erreichen. Die Ursache dafür ist die angestrebte Energieeffizienz der DSPs, die mit GPUs nicht zu erreichen ist. Moderne GPUs benötigen mehrere hundert Watt. Leistungsschwächere GPUs für den mobilen Einsatz sind häufig auf den Hauptplatinen (Intel GMA) oder in System-on-Chip Designs integriert (PowerVR SGX). Leistungsfähige GPUs verfügen über mehrere unabhängige DMA-Controller, so dass eine sehr schnelle Speicheranbin-

dung möglich ist. Im Vergleich zu DSPs sind bei GPUs auch das Speichervolumen und der Speichertakt deutlich größer.

Die sehr effiziente Berechnung von Matrizen in Fließkomma-Genauigkeit und eine sehr schnelle Anbindung an externen Speicher sind die ausschlaggebenden Argumente für die erweiterte Verwendung von GPUs als parallele Datenprozessoren für Nicht-Graphik-Anwendungen. Dies wird insbesondere dadurch begünstigt, dass die strenge Pipeline-Architektur aus Abb. 4.3 von der Industrie [90] aufgegeben wird. Die Ursache dafür hängt mit den systembedingten Nachteilen der Pipeline-Architektur zusammen. Zum Beispiel können im klassischen Ansatz keine Zwischenergebnisse aus den Pipelinestufen in den Speicher geschrieben werden, so dass keine Möglichkeit besteht, mehrfach redundante Berechnungen zu vermeiden.

Das mit dem Jahr 2008 neu eingeführte Hardwarekonzept des Unified-Shader ist an die Philosophie der Multi-Core-CPUs angelehnt. Die strenge Pipeline der unterschiedlichen Shader wurde durch eine Ansammlung von Fließkomma-Prozessoren ersetzt, die die beliebigen Berechnungsschritte der alten Pipeline ausführen können. Ein Hardwarescheduler verteilt die Aufgaben, je nach Auslastung, auf die Prozessoren, mit dem Vorteil, dass der Compiler davon entlastet wird.

Dieser Entwicklung folgend, sind GP-GPUs (General-Purpose-GPU) speziell für den High-Performance-Computing-Sektor (HPC) auf den Markt gebracht worden [75]. In ihrer Charakterisierung nähern sich die GP-GPUs den CPUs noch weiter an, als es bei den neuen GPUs mit Unified-Shader ohnehin schon der Fall ist. Traditionelle GPUs besitzen keinen Cache, so dass bereits einfache Berechnungen aufgrund der hohen Latenzzeiten des externen Speichers sehr schnell ineffizient werden. GP-GPUs besitzen hingegen einen erweiterten L1- und L2-Cache. Einfache Bit-Operationen, wie sie von SSE-Befehlen bekannt sind, sind mit Standard-GPUs nur sehr schwer zu realisieren. GP-GPUs hingegen habe diese Funktionen ebenfalls nativ in Hardware implementiert. Um den besonderen Anforderungen des High-Performance-Computing besser gerecht zu werden, verfügen GP-GPUs über ECC-gesicherten Speicher. Mit Hilfe fehlererkennender Codes und einer Reduzierung des Speichertaktes kann die Anzahl falsch übertragener Datenwörter reduziert werden. In klassischen GPUs werden einzelne korrumpierte Datenwörter hingegen toleriert.

Hybride Systeme, die in ihrer Komplexität, Energieaufnahme und Spezialisierung zwischen CPU und GP-GPUs angesiedelt werden müssen, sind Architekturen wie zum Beispiel IBMs Cell.B/E oder ClearSpeed Prozessoren. Sie sind mit der Motivation entstanden, Standard-CPUs durch eine Multi-Core-Architektur zu beschleunigen. Sie besitzen ebenfalls eine relativ hohe Anzahl an Prozessorkernen auf einem Chip, sind jedoch im Unterschied zu GPUs vielseitiger einsetzbar. Denn ein gravierender Nachteil der GPUs liegt darin, dass nicht jede scheinbar für GPUs geeignete Applikation in der Lage ist, deren parallele Ressourcen auch optimal auszunutzen. Insbesondere eine ungeeignete Form des Speicherzugriffs sowie GPU-untypische Datenbreiten bzw. -präzisionen können den anfänglichen Zeitgewinn, der durch die vielen parallelen Berechnungspfade zustande kommt, schnell wieder reduzieren. Zusammenfassend lässt

sich sagen, dass aufgrund der Entwicklung bei den GPUs und den vorgenommenen Änderungen, die zu den GP-GPUs führen, die Hochleistungs-CPUs eine ernst zunehmende Alternative erhalten haben.

4.2. Programmierbare Hardware

Alle bisher vorgestellten Hardwarearchitekturen bestehen aus fest definierten Schaltkreisen, für die entweder der Hersteller oder ein Drittanbieter Softwareschnittstellen und Compiler zur Verfügung stellt. Der Anwender kann üblicherweise durch die Benutzung einer weit verbreiteten Hochsprache, wie zum Beispiel einem C-Derivat, die Hardware zur Lösung der Applikation verwenden. Dies gilt insbesondere für Standard- und Hochleistungs-CPUs, die mit Hilfe spezieller Software-Schnittstellen-Aufrufe programmiert werden können. GPUs verwenden ebenfalls eine graphikkartenspezifische C-Erweiterung, die hersteller-abhängig ist. Eine hardware-unabhängige Standardisierung der Programmiersprachen für GPU, DSP und CPU ist die OpenCL-Softwarearchitektur, wobei zu beachten ist, dass die volle Ausbeutung der Leistungsfähigkeit durch jede zusätzliche Abstraktionsebene erschwert wird.

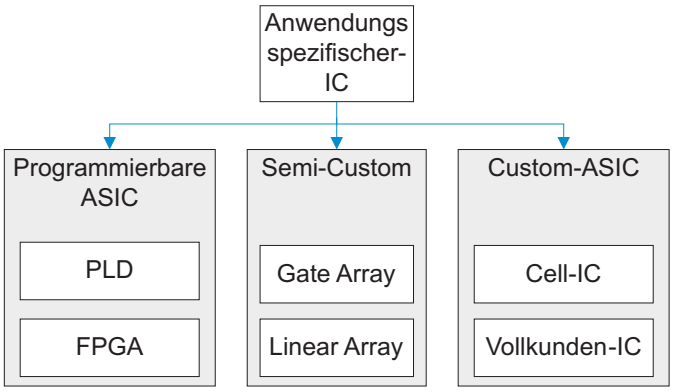
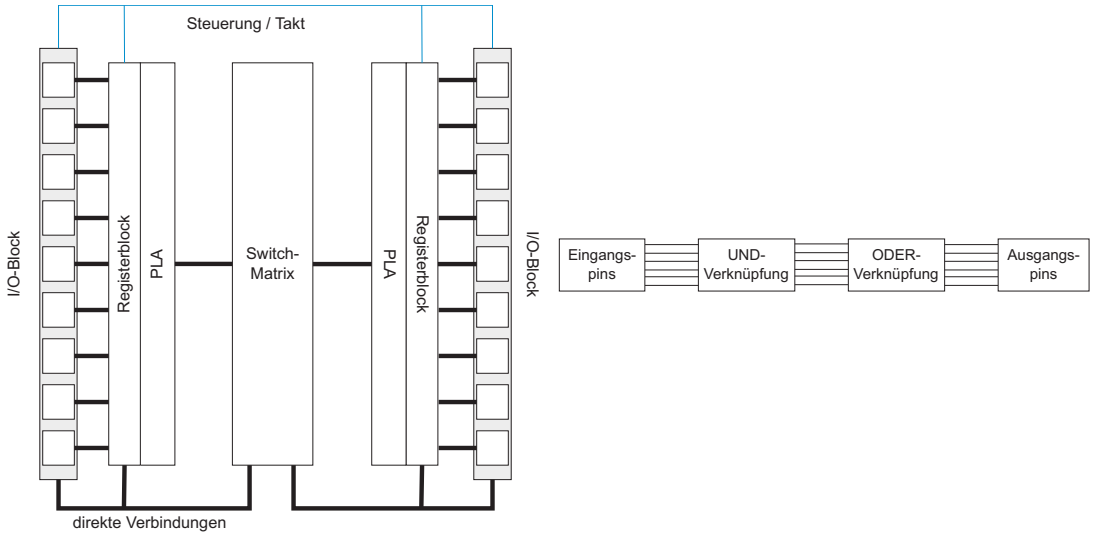


Abbildung 4.4.: Anwendungsspezifische Schaltkreise

Einen größeren Freiheitsgrad erlauben Hardwarearchitekturen, die eingangs nicht vollständig definiert oder verdrahtet sind bzw. die komplett nach Kundenwunsch gefertigt werden. In anwendungsspezifischen Schaltkreisen (ASICs) (Abb. 4.4) besteht abhängig von der gewählten Technologie zum Beispiel die Möglichkeit, das Verhältnis zwischen Logik und On-Chip-Speicher optimal zu bestimmen. Die Realisierung der Schnittstellen und die Datenwortgröße können ebenfalls den Anforderungen exakt angepasst werden, so dass die Ausnutzung der zur Verfügung stehenden Ressourcen maximal ist. Vollkunden-ICs werden vollständig nach Kundenwunsch im Werk hergestellt, so dass die Funktion der analogen und digitalen Baugruppen sowie deren Verdrahtung vom Anwender nicht mehr geändert werden kann.

Gate-Arrays hingegen erlauben die Festlegung der Logik beim Anwender. Wafer-Rohlinge mit festen Strukturen werden in großer Zahl beim Chip-Hersteller produziert, die

anschließend beim Anwender durch die Aufbringung zusätzlicher Verbindungsschichten in ihrer Funktion definiert werden. Durch diese Strategie verringert sich der üblicherweise lange Entwurfs- und Fertigungsprozess, wie er im Vollkunden-Segment gegeben ist. Während im Semi-Custom-Prozess der Anwender nach wie vor in der Chipherstellung involviert ist und Technologie bereithalten muss, kann er im programmierbaren ASIC-Bereich darauf verzichten. Dies ist insbesondere für Forschung und Entwicklung interessant, wo die hohen Fixkosten sowohl für Semi- als auch Full-Custom-ICs nicht aufwendbar sind und auch nicht durch hohe Stückzahlen kompensiert werden können. Programmable Logic Devices (PLDs), complex PLDs (CPLD) und Field Program-



(a) Architektur eines CPLDs vom Typ Xilinx[1], Coolrunner II.

(b) Programmierbare logische Anordnung (PLA). Die UND- und ODER-Gatter sind frei programmierbar.

Abbildung 4.5.: Verbindungsmatrix [3]

mable Gate Arrays (FPGAs) gehören zur Kategorie der programmierbaren Schaltkreise. Sowohl die Auswahl der Logik-Funktionen als auch die Verdrahtung werden vom Anwender vorgenommen. Eine relativ einfache programmierbare Schaltung wird durch zum Beispiel ein Programmable Logic Array (PLA) realisiert. UND- und ODER-Verknüpfungen sind frei kombinierbar (Abb. 4.5b).

CPLDs sind eine Weiterentwicklung der PLDs, bei denen eine Vielzahl von PLAs als Makros in Matrixform auf dem Chip aufgebracht werden, die über Signalpfade miteinander verbunden sind. Durch die programmierbare Verbindungsmatrix, die auch die Ein- und Ausgabepins des Chips mit den PLAs verbindet, können komplexere Schaltungen realisiert werden, als es mit einfachen PLDs möglich ist. Durch Erweiterung der PLAs mit Registern können die rein kombinatorischen Schaltungen um sequentielle Funktionalität erweitert werden. Abb. 4.5a ist eine schematische Darstellung eines

[1] `www.xilinx.com`

CPLDs vom Typ Xilinx CoolRunner II. FPGAs funktionieren ähnlich, jedoch werden

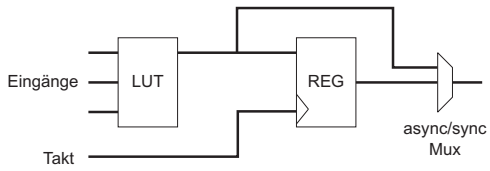


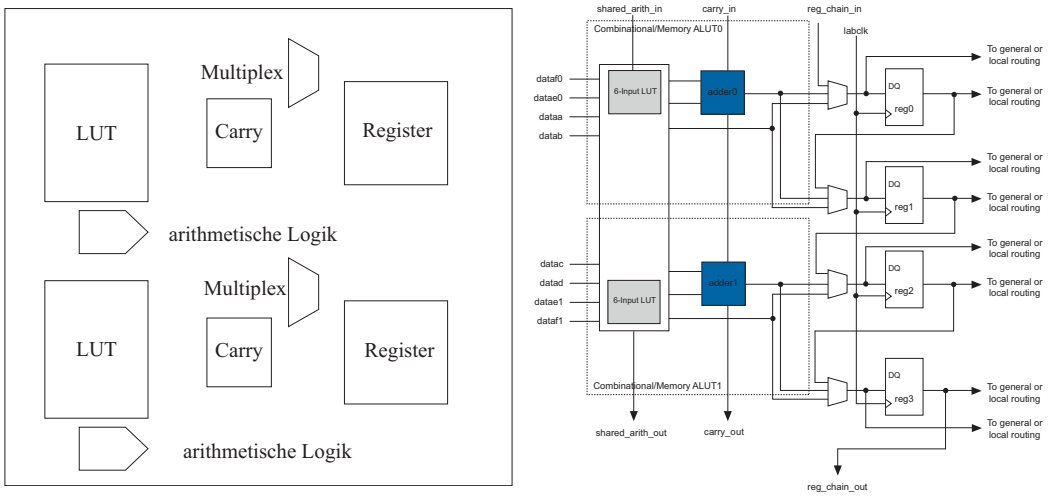
Abbildung 4.6.: LUT-basierte Logikzelle

die Logikelemente bzw. Zellen (Abb. 4.6) in Form von Look-up-Tabellen (LUT) oder Multiplexern realisiert, die von Registern umgeben sind. Die Eingänge der Tabelle bilden die Adresse auf den Tabelleneintrag bzw. die Speicherzelle. Mit Hilfe der frei programmierbaren Tabellen kann jede logische Binäroperation entsprechend der Anzahl der Eingänge abgebildet werden. Die LUT können durch Konfiguration zusätzlich entweder als Schieberegister oder als Speicher (RAM) verwendet werden. Mehrere Tabellen, zusätzliche Logikgatter für Addition/Subtraktion, Übertragslogik und Register werden üblicherweise in Blöcke zusammengefasst, die je nach Hersteller unterschiedliche Namen haben können. Dazu zählen sogenannte Slices für Xilinx FPGAs bzw. Adaptives-Logik-Modul für Altera[2] FPGAs. Die Kombination mehrerer Logikzellen zu einem Block hat den Vorteil, dass für kurze Verbindungen innerhalb des Blocks einfache Treiber und Schaltpunkte verwendet werden können. Zum einen wird dadurch Fläche eingespart und zum anderen wird die Signallaufzeit verkürzt. Zusätzlich zur Flächeneinsparung und der Verkürzung der Signallaufzeiten werden die globalen Verdrahtungsressourcen entlastet, wenn stark zusammenhängende Logikzellen auf die Ressourcen in einem Logikblock abgebildet werden können.

Das optimale Verhältnis von Logikzellen pro Block mit Hinblick auf den Flächenverbrauch ist abhängig von der verwendeten Größe der Funktionsgeneratoren und der Fertigungstechnologie. Tabellen mit drei bis vier Eingängen haben sich als optimal bezüglich des Flächenverbrauchs herausgestellt [1]. Vier bis sechs Eingänge sind hinsichtlich der maximalen Taktrate optimal, was mit der verkürzten Logiktiefe und der davon abhängigen Signallaufzeit zusammenhängt. Der Nachteil der auf Geschwindigkeit optimierten LUTs besteht darin, dass ein Teil der FPGA-Fläche ungenutzt bleibt, wenn einfache Funktionen auf komplexe Tabellen abgebildet werden. Ein Kompromiss besteht darin, wahlweise zwei LUTs mit relativ wenigen Eingängen zu einem komplexeren LUT adaptiv zu kombinieren. Altera FPGAs besitzen aus diesem Grund zwei 3-LUTs und zwei 4-LUTS, die in einem Logikblock zusammen gefasst sind (Abb. 4.7).

Zwischen den einzelnen Logikblöcken gibt es ebenfalls lokale Verbindungen zu Nachbarblöcken, die dazu geeignet sind, mehrere Logikblöcke zusammenzuschalten. Hierdurch können komplexere arithmetische und logische Funktionen realisiert werden. Außer über die kurzen lokalen Verbindungen, ist jeder Logikblock an ein Segment eines globalen Schaltnetzwerks angeschlossen. Die programmierbare Schaltmatrix ver-

[2] `www.altera.com`

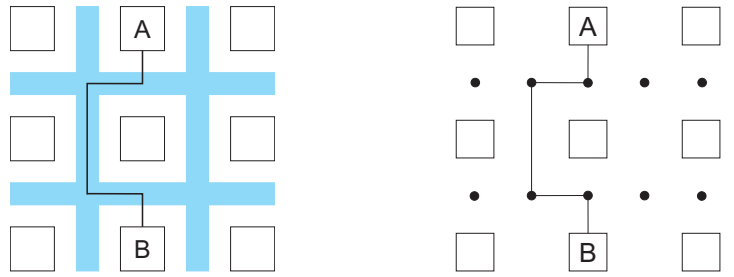


(a) Stark vereinfachtes Schema eines Logik- und Registerblocks

(b) Schema eines Logik-/Registerblocks am Beispiel des Altera Stratix 5 (ALM)

Abbildung 4.7.: Look-Up-Tabelle, arithmetische Logik und Register zusammengefasst

bindet die zwei weiteren Hauptkomponenten eines FPGAs miteinander. Das sind die bereits erwähnten programmierbaren Funktionsblöcke und die programmierbaren Ein- und Ausgänge. Der Hauptunterschied zu CPLDs besteht im Wesentlichen in der besseren Vorhersagbarkeit der Signallaufzeiten auf Seiten der CPLDs. Dies geht auf die insgesamt durchgehende Verdrahtung der CPLDs zurück. Die Schaltmatrix der FPGAs ist typischerweise hierarchisch bzw. segment-orientiert, wodurch eine endgültige Bestimmung der Signallaufzeiten erst im Place-und-Route-Schritt möglich ist (Abb. 4.8). Die Technologie der FPGAs basiert vorrangig auf SRAM- oder Flash-Speicherzellen.



(a) CPLD Verbindungsmatrix mit durchgehenden Verbindungen

(b) Segmentbasierte FPGA Verbindungsmatrix

Abbildung 4.8.: Verbindungsmatrix [3]

Anti-Fuse-FPGAs finden ihre Anwendung hauptsächlich in einer strahlungsbelasteten Umgebung und nehmen damit eine Nebenrolle ein. Die SRAM- oder Flash-Zellen die-

nen als Speicher zum einen für den Zustand der Look-up-Tabellen und zum anderen für den Zustand der Schaltpunkte in der Verbindungsmatrix. Die in CMOS kostengünstig gefertigte SRAM-Speicherzelle verliert ihren Speicherzustand mit Verlust der Betriebsspannung. Erst mit der Wiederkehr der Spannung nimmt die Speicherzelle einen nichtdeterministischen Zustand an. Dadurch, dass der Zustand der Speicherzelle willkürlich ist, muss der FPGA nach dem Einschalten konfiguriert werden. Häufig wird dazu die Konfiguration (Bitfile) aus einem externen PROM in den FPGA geladen. Non-volatile-Flash- oder Anti-Fuse-FPGAs besitzen an dieser Stelle einen Vorteil. Die gleiche Programmiervoraussetzung gilt für CPLDs, die es auch auf SRAM-Basis gibt. Mittlerweile sind In-System- oder Multi-Die-FPGAs erhältlich, die SRAM und Flash in einem Gehäuse integrieren, wodurch der externe Verdrahtungsaufwand auf der Leiterplatte minimiert wird. Zu beachten ist, dass die Multi-Die-FPGAs nicht ähnlich strahlungstolerant wie Anti-Fuse-FPGAs sind. Neben den klassischen Optimierungskriterien Fläche und Geschwindigkeit, hat in den letzten Jahren aufgrund der Fortschritte in der Integration und der Mobilität der Faktor Energieverbrauch an Bedeutung gewonnen. Mit zunehmender Anzahl an Transistoren pro Fläche und zunehmender Geschwindigkeit hat auch der Anteil der Verlustleistung zugenommen [81]. Statische Verlustleistung wird durch Leckströme verursacht und ist vorrangig technologiebedingt, während dynamische Prozesse zur Laufzeit zum Tragen kommen.

Mitunter können zusätzlich während des Startvorgangs sehr hohe Einschaltströme auftreten. Erst mit der Programmierung wird der FPGA in einen operationalen Modus versetzt, so dass die Leistungskennlinie in der Bootphase der SRAM-FPGAs eine signifikante Spitze aufweist. Die Energieaufnahme im Betrieb des FPGAs ist von dem geladenen Design abhängig und ist hauptsächlich durch das Umladen kapazitiver Elemente bestimmt. Eine Möglichkeit, die Energieaufnahme zu reduzieren, besteht darin, für einzelne Logikbereiche den Takt zeitweise anzuhalten. Ähnliche Strategien lassen sich auch für zeitweise ungenutzte Hard-IP-Cores und FPGA-nahe Peripherie bzw. FPGA-Schnittstellen realisieren, so dass der FPGA adaptiv den Energieverbrauch steuern kann. Mit dem Ziel, soviel Programmierkapazität wie möglich dem Benutzer zur Verfügung zu stellen, werden FPGAs eingesetzt, in denen die am häufigsten verwendeten Funktionen bereits fest integriert sind. Die fest verdrahteten Hard-IP-Cores minimieren insbesondere den Verdrahtungsaufwand, der einen Großteil der Fläche eines FPGAs verbraucht [70]. Die Hard-IP-Cores können im Gegensatz zu Soft-IP-Cores vom Anwender nicht umprogrammiert werden. Lediglich eine Konfiguration bzw. Deaktivierung sind möglich. Aufgrund dessen, dass sehr viele Algorithmen größere Mengen Zwischenspeicher benötigen, wurden bald darauf RAM-Blöcke als Hard-IPs in die FPGAs integriert. Diese sind als Single- oder Dual-Port-RAM konfigurierbar. FIFOs und Schieberegister sind ebenfalls mit dem Block-RAM umsetzbar. Mit der Integration von dedizierten DSP-Blöcken wurden die FPGAs um Fließkomma-Recheneinheiten erweitert, wodurch insbesondere Hardwareentwürfe für die Signalverarbeitung profitieren.

Mit jeder neuen Generation kommerzieller FPGAs sind bereits integrierte IP-Cores verbessert und neue hinzugenommen worden, so dass heutige FPGAs geeignet sind,

vollständige Systeme auf einem Chip (System-on-Chip) abzubilden. Dazu zählen Prozessorarchitekturen und die hardwareseitige Integration schneller serieller Punkt-zu-Punkt Protokolle. Aufgrund der Vielzahl unterschiedlicher Hard-IP-Cores und Fertigungstechnologien haben sich FPGA-Familien entwickelt, die jede für sich eine Spezialisierung darstellen. Der Schwerpunkt liegt zumeist auf Anwendungen mit entweder viel Logik-, Kommunikations- oder Arithmetikbedarf.

4.3. Benchmarks

Die unterschiedlichen Implementierungsstrategien und Fertigungsprozesse der Hersteller erschweren einen direkten Vergleich der PLDs untereinander. So hat insbesondere die Schaltmatrix innerhalb der FPGAs einen großen Einfluss auf die Leistungsfähigkeit des Entwurfs [62]. Je mehr Schaltpunkte bzw. Segmente für die Verdrahtung eines Signals verwendet werden müssen, desto mehr verzögern sich die Signallaufzeiten. Ein Entwurf ist unter diesen Umständen möglicherweise nicht mit der erforderlichen Taktrate zu betreiben bzw. ist im schlimmsten Fall in der gewählten Form nicht realisierbar. Anfänglich wurde die aus der ASIC-Entwicklung stammende Kenngröße der Logikgatter als Kriterium für die Logikressourcen angegeben. Sie beschreibt die Anzahl der realisierbaren NAND-Gatter in Standardzellen-ASICs und kann als ein Maß für die Komplexität eines Entwurfs verwendet werden. Die am verbreitesten rekonfigurierbaren Hardwarebausteine verwenden wiederum Tabellen zur Erzeugung der Logik, wodurch ein großer Ermessensspielraum bei der Angabe der realisierbaren Gatteranzahl gegeben ist. Stattdessen wird oftmals von den Herstellern die Anzahl der vorhandenen Logikzellen (Abb. 4.6) angegeben, so dass zumindest ein Vergleich innerhalb einer FPGA-Familie eines Herstellers möglich ist [3].

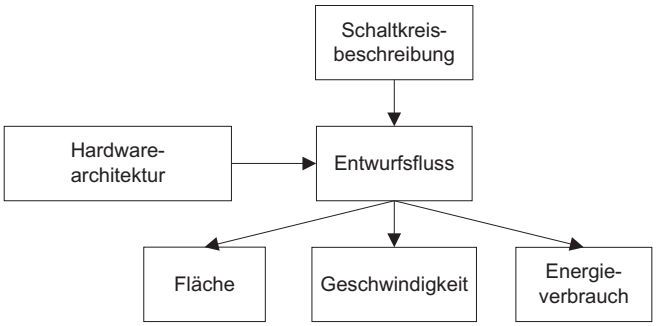


Abbildung 4.9.: Benchmarktest unterschiedlicher Hardwarearchitekturen empirisch ermittelt [71]

Nichtsdestotrotz ist diese Angabe willkürlich und nicht normiert, so dass es sinnvoller ist, auf hardwarenahem Niveau die Grundbausteine eines PLDs zu vergleichen. Für FPGAs kann als Basis zum Beispiel die Anzahl und Struktur der Register, Block-

[3] Die Angabe äquivalenter Logikzellen in Xilinx FPGAs beinhaltet zusätzlich die Multiplex- und arithmetischen Einheiten (Abb. 4.7a)

RAMs, Taktgeneratoren und Netze herangezogen werden. Da die Leistungsfähigkeit jeder spezialisierten Hardware abhängig von der Zielapplikation ist und die Hardware für eine manuelle Betrachtung ohnehin zu komplex ist, existiert eine Reihe von Benchmarks IWLS2005 [2], AlteraSet [4], OpenCores + TorontoSet [70] und weitere, die die verschiedenen Aspekte des Schaltkreisentwurfs bewerten. Die Standardmetriken Flächenverbrauch, Timingverhalten, Energieeffizienz und die Leistungsfähigkeit der Entwurfswerkzeuge [71],[89] werden zum Teil durch anwendungsspezifische Merkmale erweitert [20], [16]. Zur Ermittlung des relativen Flächenverbrauchs einer Architektur wird üblicherweise auf eine Basis, die zum Beispiel der Flächenverbrauch einer Logikzelle sein kann, normiert. Jedoch hat sich hierfür keine Standardbasis als Grundlage etabliert.

Alternativ zu den quantitativen Analysen mit Hilfe der Entwurfswerkzeuge, können Architekturen auch von einem theoretischen Standpunkt aus untersucht werden. Dazu wird üblicherweise der Benchmarkalgorithmus in eine deklarative Form gebracht, so dass eine Trennung zwischen Problembeschreibung und Problemlösung möglich ist [21]. Anschließend wird die Problembeschreibung in Form eines Signalflussgraphen solange in kleinere Operationen zerlegt, bis das Entwurfsmodell Realisierungen für die Operation kennt. Anschließend werden für die realisierbaren Operationen die Werte für die Metriken wie zum Beispiel Flächenverbrauch, Timing (Routing), Energieverbrauch ermittelt. Der Gesamtflächenverbrauch errechnet sich zum Beispiel aus der Summe der einzelnen Hardwarekomponenten.

4.4. Plattformen im Vergleich

Mit der Einführung der Multi-Core-CPUs und den Veränderungen bei den GPUs sind neben den PLDs weitere Zielplattformen verfügbar geworden, die aufgrund ihrer Vielzahl an parallelen Ressourcen zur Hardwarebeschleunigung von daten- und rechenintensiven Anwendungen eingesetzt werden können. Veröffentlichungen und Untersuchungen aus der Vergangenheit lassen mehrere allgemeine Aussagen über die Leistungsfähigkeit der vorgestellten Zielplattformen zu, die im Folgenden wiedergegeben werden, wobei deren heuristischer Charakter an dieser Stelle betont werden soll. K. Asanovic et al. [6] entwickelten eine unscharfe Einteilung einer Menge von Anwendungen in Lösungsklassen, deren Berechnungsvorschriften Ähnlichkeiten zueinander besitzen. Dabei geht es vorrangig darum, die Anforderungen bzw. Limitierungen hinsichtlich Ablaufsteuerung, Speicher und Berechnungsaufwand für die jeweilige Klasse zu benennen, um daraus eine Zielplattform abzuleiten.

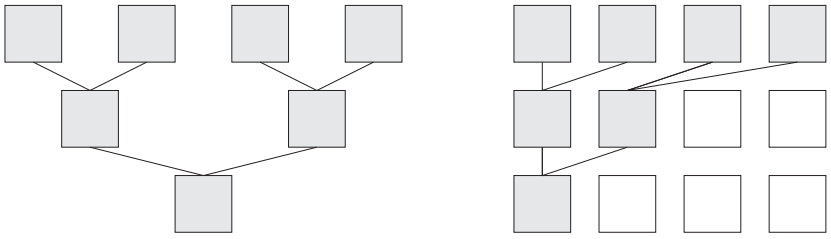
Traditionell eignen sich Algorithmen für eine FPGA-Implementierung, wenn sie folgende Eigenschaften aufweisen [10], [38]:

- Der Algorithmus verwendet vorrangig ganzzahlige Datentypen vom Typ Integer. Insbesondere wenn die binäre Darstellung kein Vielfaches von einem Byte ist, ist eine effiziente Umsetzung mit Hilfe reprogrammierbarer Hardware möglich. Diese Eigenschaft ist zentral und ermöglicht einen signifikanten Vorteil gegenüber

Byte-orientierten Plattformen.

- Die arithmetischen Operationen haben nur eine geringe Komplexität. Die Logikblöcke der FPGAs besitzen dedizierte Logik (Abb. 4.7a) zur Realisierung einfacher arithmetischer Operationen (Addition, Multiplikation).
- Der Algorithmus besitzt tendenziell keine rekursiven Abhängigkeiten.
- Applikationen weisen eine hohe Daten-Parallelität auf.
- Die Ablaufsteuerung ist zur Laufzeit relativ statisch.

Ein klares Unterscheidungsmerkmal zwischen PLDs und GPUs, GP-GPUs bzw. CPU-Realisierungen ist die Flexibilität der PLDs in Bezug auf Operationen, die auf Bitebene ausgeführt werden. Ebenso ist es mit GPUs und mit CPUs nur bedingt möglich, auf die in parallelen Anwendungen häufig vorkommende Datenreduktion flexibel zu reagieren. In Abb. 4.10b stellen die weißen Felder unbenutzte, jedoch blockierte Ressourcen dar, so dass in GPU- und CPU-Implementierungen hierauf besondere Rücksicht genommen werden muss. Erfahrungsgemäß ist es sehr schwierig, die Prozessoren der GPU voll auszulasten. Dies hängt insbesondere mit der Latenz beim Laden von Befehlen und mit der Latenz sowie Bandbreite von Speicherzugriffen zusammen. Im Idealfall können die Daten ohne Unterbrechung in den FPGA kopiert werden, da die Speicherbefehle in Hardware codiert sind. Der Entwickler hat an dieser Stelle die Möglichkeit, die Latenz für den Speicherzugriff durch Pipelining der Befehle zu minimieren. Dies ist für GPU bzw. CPU ungleich schwieriger zu erreichen.



(a) PLD (b) GPU

Abbildung 4.10.: Datenreduktion für GPU- und PLD-Plattformen [14]

Im Gegenzug dazu sind die GPUs der neusten Generation für wissenschaftliche Aufgaben im Double-Precision-Bereich prädestiniert. FPGAs können ebenfalls mit Fließkommazahlen umgehen, jedoch bei weitem nicht in dem Umfang wie die GPUs. Des Weiteren ist die Programmiermethodik für GPUs bzw. CPUs an der klassischen Softwareentwicklung orientiert. Softwarebibliotheken stellen bereits implementierte Schnittstellen zu den Ressourcen her. Dadurch ist es möglich, auf Funktionen zum Beispiel für den Speichertransfer zurück zu greifen. Aufgrund der sich oftmals ändernden Hardwaregrundlage müssen FPGA-Speicherkontroller häufig neu implementiert werden. Die Folge ist eine hohe Fehleranfälligkeit und damit verbundene hohe Entwicklungszeit. Die

hohen Kosten und Komplexität der Hardwareentwicklung gelten somit als Haupthindernis für eine Realisierung. Die Fortschritte und Ergebnisse auf dem Gebiet der Hardwareprogrammierung sollen im folgenden Abschnitt vorgestellt werden, wobei zwei wesentliche Ansätze zu erkennen sind. Zum einen wird versucht, die etablierten Hochsprachen der Softwareentwicklung mit Funktionen anzureichern, die die Hardware stärker berücksichtigen, sodass aus diesen eine synthetisierbare Hardwarebeschreibung automatisch generiert werden kann. Mit diesem Ansatz soll ein besserer Zugang für klassische Softwareentwickler zur Hardwareentwicklung geschaffen werden. Ein zweiter und in gewisser Weise gegensätzlicher Ansatz orientiert sich an der traditionellen Hardwareentwicklung mit Hilfe der Hardwarebeschreibungssprachen VHDL und Verilog. Diese werden um höhere Abstraktionsebenen ergänzt, wobei es jedoch vorrangig darum geht, durch verbesserte Simulations- und Verifikationsmethoden die Entwicklungszeit und -kosten zu verkürzen. Die Arbeiten auf dem Gebiet des Hardware-/Software-Codesigns befinden sich in einem relativ frühen Stadion und verbinden beide Entwicklungsansätze miteinander, die in einer integrierten und automatisierten Entwicklung von Soft- und Hardware für heterogene Systeme münden sollen.

4.5. Entwurf heterogener Systeme

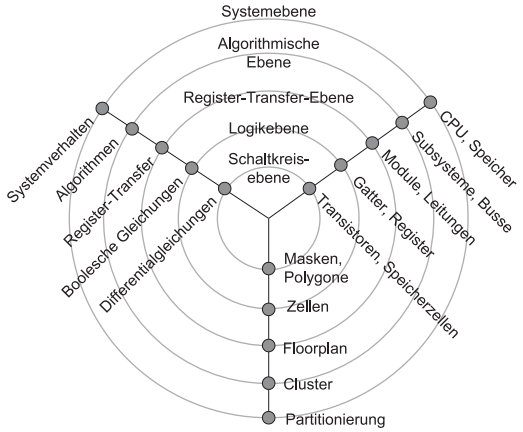
Heutige Systeme sind vorwiegend heterogene Systeme. Darunter werden Systeme verstanden, in denen die unterschiedlichsten Hard- als auch Softwarekomponenten auf optimale Art und Weise zusammenarbeiten. Der Grund dafür ist der, dass die heterogenen Systeme sich am effizientesten an die durch die Applikation gestellten Anforderungen anpassen lassen. Der Entwurf heterogener Systeme ist somit nicht nur auf Hardware oder nur auf Software beschränkt [68]. Dies führt zwangsläufig zu einer sehr komplexen Systembeschreibung, denn die heute zur Verfügung stehende Hardware ist selber heterogen. Sowohl Analog- als auch Digitaltechnik, Aktoren und Sensoren, passive und aktive Komponenten müssen von der Systembeschreibung erfasst werden. Dabei wird der Entwurf heterogener Systeme von der Systembeschreibung geleitet, mit dem Ziel, alle Programme für die zu programmierenden Komponenten daraus zu erzeugen.

Grundsätzlich gibt es drei wesentliche Aspekte, die für den Entwurf heterogener Systeme von Bedeutung sind. Das ist zum einem die bereits angedeutete Systembeschreibung. Der zweite Aspekt behandelt die Systempartitionierung und beinhaltet die Zerlegung und Zuordnung der Anwendung auf die einzelnen Teile des heterogenen Systems, sowie die Kommunikation zwischen den Teilen. Im letzten Punkt wird die Verifikation thematisiert, die ebenfalls eng mit den Randbedingung der Applikation und Hardware verknüpft ist.

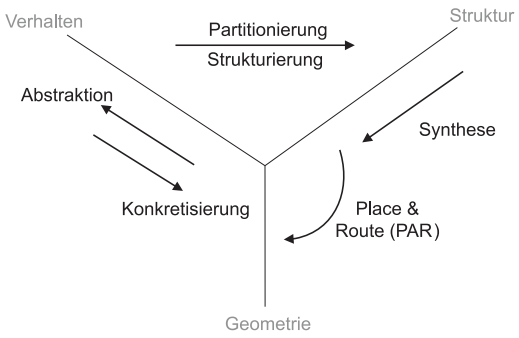
Von der Systembeschreibung wird erwartet, dass sie zwei widersprüchlichen Anforderungen gerecht wird. Dazu zählt, dass das Programmiermodell vom Entwickler intuitiv benutzbar ist, indem von den Details der zugrundeliegenden Hard- und Softwarekomponenten abstrahiert wird, so dass der Entwickler gar nicht bzw. erst relativ spät mit der eigentlichen Realisierung konfrontiert wird. D. h. der Entwickler sollte von den

komplexen Randbedingungen und Möglichkeiten der heterogenen Systeme weitestgehend befreit sein. Die frühzeitige Verifikation auf Systemebene birgt zusätzliche Vorteile. Mit Hilfe der Systembeschreibung muss es jedoch möglich bleiben, einen Entwurf zu realisieren, der der Hardware optimal angepasst ist, um letztendlich die Leistungsfähigkeit der Hardware voll auszuschöpfen. Durch Verallgemeinerung der Hardware wird dies jedoch erschwert.

Die Zusammenhänge zwischen den Abstraktionsebenen können gut durch das Y-Diagramm von D. D. Gajski [30] in Abb. 4.11a dargestellt werden. Die drei wesentlichen Aspekte Geometrie, Verhalten und Struktur der verschiedenen Abstraktionsebenen spiegeln sich in den Ästen des Diagramms wieder. Die Verhaltenssicht beschreibt die Funktionsweise des Systems, die von der Geometriesicht letztendlich realisiert wird. Letztere hat nur Bedeutung für den Entwurf von zum Beispiel Voll-Kunden-ICs. Für PLDs ist nach dem Place-und-Route-Schritt Schluss (Abb. 4.11b). Der Entwurfsprozess sieht typischerweise vor, das Verhalten des heterogenen Systems auf höchster Ebene zu beschreiben. Durch mehrfache Konkretisierung und Transformation in die Struktursicht, wird das Verhaltensmodell schrittweise verfeinert. Mit Hilfe von Synthesewerkzeugen ist es möglich, Teilmodelle aus der Struktursicht in die Logikebene maschinell zu überführen.



(a) Abstraktionsebenen nach D. D. Gajski [30]



(b) Entwurfsprozess im Y-Diagramm [30]

Abbildung 4.11.: Y-Diagramm von D. D. Gajski [30]

Für klassische Prozessoren existieren seit langem Compiler, die aus Verhaltensbeschreibungen auf der System- bzw. algorithmischen Ebene automatisch in einen Maschinencode übersetzen. Das zeitliche Verhalten des Systems wird von dem Compiler durch die sequentielle Abarbeitung der Eingabe aufgelöst. Schwieriger wird es, wenn das System die Konzepte für eine parallele Abarbeitung realisieren soll. Es spielt dabei keine wesentliche Rolle, ob es sich um SIMD-Prozessoren oder rekonfigurierbare Hardware handelt, für die der Compiler Programme erzeugen soll. Die Randbedingungen für rekonfigurierbare Hardware sind zwar komplexer, aber das Hauptproblem für einen vollautomatisch parallelisierenden Compiler besteht darin, die parallelisierbaren Teile

zu identifizieren.

Auf dem Weg hin zu diesem Compiler haben sich Spracherweiterungen auf Basis von Standardsprachen wie C und Fortran (UPC, OpenMP) entwickelt, die zumindest eine manuelle Beschreibung der Parallelität ermöglichen. Die Compiler sind vorwiegend auf Prozessoren, DSPs und GPUs bzw. GP-GPUs beschränkt. In ähnlicher Weise sind Ansätze vorgestellt worden, die insbesondere den Entwurf rekonfigurierbarer Hardware berücksichtigen und ihren Ursprung in der Hardwareentwicklung haben. Dazu zählen die bekannten Vertreter Handel-C, Mitrion-C, Impulse-C und FPGA-C, die alle eine C ähnliche Syntax besitzen. Dabei geht es vorrangig darum, die bestehenden Algorithmen und Konzepte aus der Softwareentwicklung zu übernehmen. Mit dem Schwerpunkt in Richtung Signalverarbeitung existieren grafisch orientierte Systembeschreibungen wie MatLab/Simulink und Xilinx System Generator. Am vielversprechendsten scheint jedoch die Entwicklung im Hinblick auf die Sprache SystemC (eigentlich eine C++-Klassenbiblothek, IEEE 1666), da diese von den dominierenden Entwicklungs- und Simulationswerkzeugen unterstützt wird und als Ergänzung zu den Hardwarebeschreibungssprachen (HDL) eingesetzt wird (Abb. 4.12).

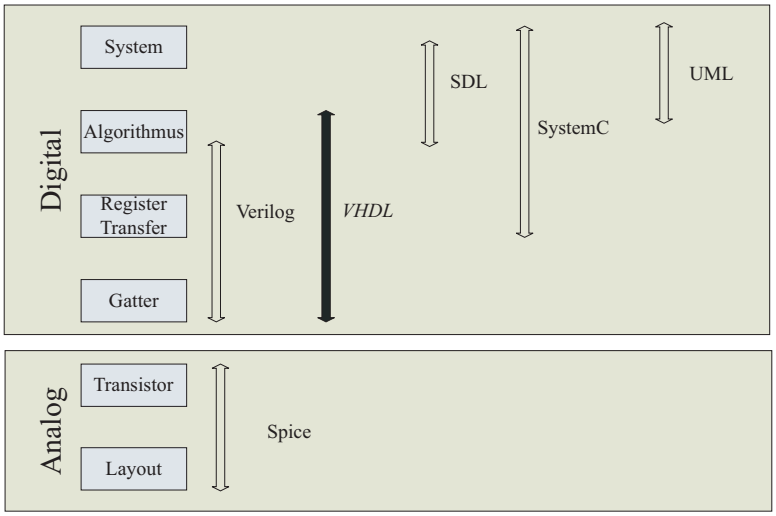


Abbildung 4.12.: Modellierungssprachen und deren Abstraktionsgrad (nach [68])

Hardwarebeschreibungssprachen wie VHDL und Verilog haben andererseits ihren Ursprung im Schaltkreisentwurf und wurden vorrangig zu Modellierung und Simulation komplexer Systeme entwickelt. Möglich ist heute nicht nur die Beschreibung und Verifikation auf der algorithmischen, Register-Transfer- (RTL) und Logikebene, sondern auch die maschinelle Transformation zwischen den Ebenen. Sowohl die Struktur- als auch die Verhaltensbeschreibung dürfen bei VHDL nebeneinander stehen. Der wesentliche Vorteil von VHDL besteht jedoch darin, dass im Unterschied zu Standardprogrammiersprachen VHDL von sich aus in der Lage ist, Nebenläufigkeit zu modellieren und zu verifizieren. Dies wird häufig als Nachteil für die Erlernbarkeit von Hardwa-

rebeschreibungssprachen betrachtet. Der Grund dafür liegt im möglicherweise schwer überschaubaren Programmablauf, der ganz natürlich durch das parallele Strukturmodell bedingt ist. Als Folge davon wird ein Reengineering erschwert und die Fehleranfälligkeit und die Entwicklungszeit werden vergrößert.

4.6. Das Hardwarebetriebssystem

Eine Möglichkeit, die Fehleranfälligkeit und Entwicklungszeit mit Hardwarebeschreibungssprachen zu minimieren, besteht in der Wiederverwendung bereits verifizierter und eventuell optimierter Komponenten. Dieses Vorgehen ist in der Domäne der Softwareentwicklung seit langem üblich und hat sich bewährt. Vorgesehen ist dafür das Konzept eines Betriebssystems, das durch Kapselung eine standardisierte Schnittstelle zu den Ressourcen ermöglicht (Abb. 4.13). Zum einen wird dadurch die Ressource austauschbar und zum anderen können die Anwendungen beliebig ersetzt werden. Eine Wiederverwendung von Systemteilen tritt demnach sowohl bei den Betriebssystemmodulen als auch bei den Anwendungen auf. Das Betriebssystem verwaltet zugleich die Ressourcen und stellt diese den Modulen bzw. Anwendungen zum gegebenen Zeitpunkt zur Verfügung. Das Konzept eines Betriebssystems für Hardware wurde erst-

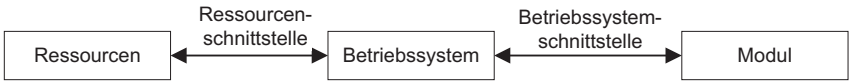


Abbildung 4.13.: Betriebssystemkonzept

malig in [68] vorgestellt und soll im weiteren Verlauf verwendet werden. Ziel ist es, eine plattform-unabhängige Hardwarebeschreibung zu erstellen, die von den Standardwerkzeugen verwendet werden kann. Zur Umsetzung des Betriebssystemkonzepts auf der Basis der Beschreibungssprache VHDL ist es erforderlich, in einer Vorstufe die Limitierungen der Beschreibungssprache VHDL durch Neuzuweisung der Signale zu umgehen[4].

Die Hardwarebeschreibungssprache VHDL kennt grundsätzlich vier Konstrukte. Die *Entity* spezifiziert die Schnittstelle eines Moduls. Ein Modul kann das vollständige System sein oder ein Teil davon. In der *Architecture* wird der strukturelle Aufbau bzw. das funktionale Verhalten des Moduls definiert. Auf diese Art und Weise wird zwischen Schnittstelle und Realisierung strikt getrennt, so dass es mehrere Realisierungen für eine Schnittstelle geben kann. Welche *Architecture* letztendlich verwendet wird, hängt von der Festlegung im *Configuration*-Teil ab. *Packages* werden verwendet, um Konstanten, wiederkehrende Datentypen und Funktionen zu definieren. Eine sequentielle funktionale Beschreibung innerhalb der Architekturen ist mit der Anweisung *Process* möglich. Strukturelle Beschreibungen müssen durch die *Component*-Deklaration eingeleitet werden und in der Schnittstellenbeschreibung mit der Komponenteninstanziierung übereinstimmen.

[4]Für Verilog gelten die Einschränkungen in ähnlicher Weise.

Die Beschreibung eines Moduls mit Hilfe von HDL setzt voraus, dass die Schnittstelle zur Übersetzungszeit vollständig definiert ist. D. h. alle Signale, die das Modul und dessen Submodule benötigen, müssen von Anfang an dem Synthesewerkzeug bekannt gemacht werden. Des Weiteren gibt es keine Möglichkeit, in VHDL über verschiedene Module hinweg globale Variablen zu definieren. In VHDL sind einzig und allein Signalzuweisungen zwischen den Modulen für diesen Zweck vorgesehen. Ein Hardwarebetriebssystem, das zum einen Hardwareressourcen kapselt und zum anderen den multiplen Zugriff auf diese verwaltet, ist auf der Anwenderseite darauf angewiesen, dass mehrere unterschiedliche Betriebsystemschnittstellen sich mit einem Betriebssystemmodul verknüpfen lassen. Auf der Hardwareseite wiederum kapselt ein Betriebssystemmodul die Anbindung an Hardwareressourcen wie zum Beispiel externen Speicher. Aufgrund der strengen Regeln für die Definition von Modulschnittstellen kann in VHDL nicht ohne weiteres eine allgemeine Betriebssystemschnittstelle für zum Beispiel Speicher erstellt werden. Damit unterschiedliche externe Hardwarekomponenten einer Kategorie (zum Beispiel SRAM, BlockRAM, DDR-RAM) dennoch mit Hilfe einer allgemeinen Schnittstelle verwendet werden können, kann ein Präcompiler eingesetzt werden, der in drei Transformationsschritten die Schnittstellenbeschreibung der Anwendung und der Betriebssystemmodule modifiziert.

Präcompiler

In einem ersten Transformationsschritt werden die Signale der Betriebssystemschnittstellen der Anwendungsteile zur Schnittstelle des Toplevel-Moduls des Projekts umgelenkt. Für die Signale der Betriebssystemschnittstellen der Betriebssystemmodule wird in gleicher Art und Weise verfahren. In einem letzten Schritt wird ein übergeordnetes Toplevel-Modul erzeugt, dass die erweiterten Anwendungs- und Betriebssystemmodule einbindet und die betriebssystemspezifischen Signale der Schnittstellen mit den Modulen verbindet. Der Strukturcompiler stellt das Bindeglied zwischen den projektbezogenen und betriebssystembezogenen Hardwarebeschreibungen (Abb. 4.14) dar. In einer Projektkonfiguration wird festgelegt, welche Betriebssystemmodule mit den Betriebssystemschnittstellen in dem Projekt verbunden werden. Im Anhang Listing A.1 ist als Beispiel eine Projektkonfiguration für ein typische Applikation angegeben. Die anwendungsspezifischen Quelldateien werden dem Strukturcompiler im Kopf der Konfiguration bekannt gemacht. `TOPLEVEL NAME` definiert die Toplevel-Entity der Anwendung. `MAIN ENTITY NAME` und `MAIN ARCHITECTURE NAME` sind die vom Strukturcompiler vergebenen Namen für die neue Toplevel-Entity bzw. Architektur, die alle hinzugefügten Signale vereint. Abhängig davon, ob das Transformationsziel eine Simulation oder eine Bitfile-Generierung ist, muss neben `OUTPUT FILE` auch der Name für die generierte UCF-Datei durch das Schlüsselwort `UCF FILE` angegeben werden.

Einschränkungen durch das Hardwarebetriebssystem

Der Strukturcompiler realisiert eine Betriebssystemumgebung durch Umleitungen gerichteter Signale. Dadurch wird die direkte Verwendung von Signalen mit dem At-

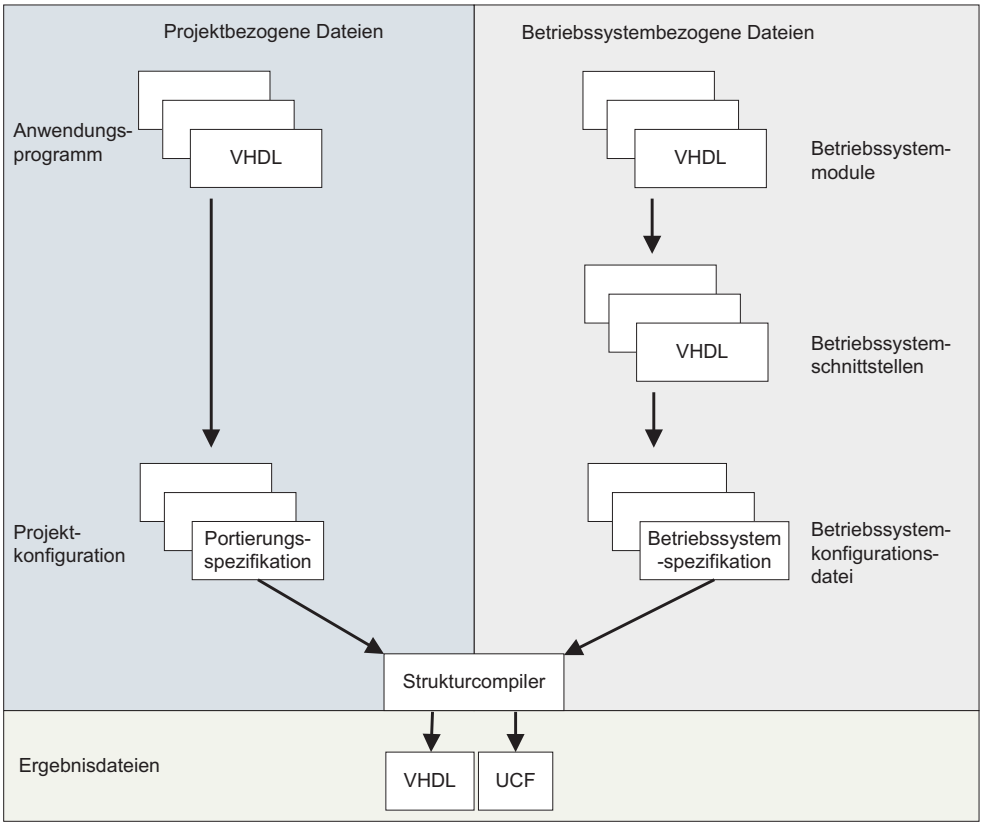


Abbildung 4.14.: Ablaufdiagramm des Präcompilers [68]

tribut `inout` verhindert. Um diese Signalform dennoch nutzen zu können, müssen diese Signal in ein IN-, OUT- und Steuersignal zerlegt werden. Eine andere weiterreichende Einschränkung betrifft den Aufruf von Betriebssystemschnittstellen innerhalb von `Generate`-Anweisungen. An dieser Stelle dürfen keine Betriebssystemkomponenten vorkommen, da der Präcompiler nicht in der Lage ist, die genaue Anzahl der Schnittstellen aufzulösen. Abschließend muss darauf hingewiesen werden, dass besondere Aufmerksamkeit bei der Zuweisung der Namen für die Betriebssystemschnittstellen erforderlich ist. Der Zusammenhang zwischen den Betriebssystemmodulen und den Betriebssystemschnittstellen wird global auf der Basis des Schnittstellennamens vorgenommen, wodurch es bei Unachtsamkeit zu falschen bzw. doppelten Zuweisungen kommen kann. Eine feiner strukturierte Zuweisung ist nicht möglich.

Kommunikationsschnittstellen

Das Hardwarebetriebssystem kennt drei unterschiedliche logische Protokolle zur Kommunikation. Dazu zählen die asynchrone Übertragung einzelnen Datenwörter, die synchrone Übertragung eines Datenwortes und die zusammenhängende synchrone Übertragung mehrerer Datenwörter. Der asynchrone Einzelwort-Transfer (Channel-

Kommunikation) wird innerhalb des Betriebssystems für die einfache Übertragung unkritischer Werte, wie zum Beispiel Startparameter verwendet (Abb. 4.15). Es existiert lediglich ein Datensignal variabler Bitbreite. Steuersignale zur Synchronisation sind nicht vorhanden, sodass der Datenempfänger die Abtastrate selber bestimmen kann.

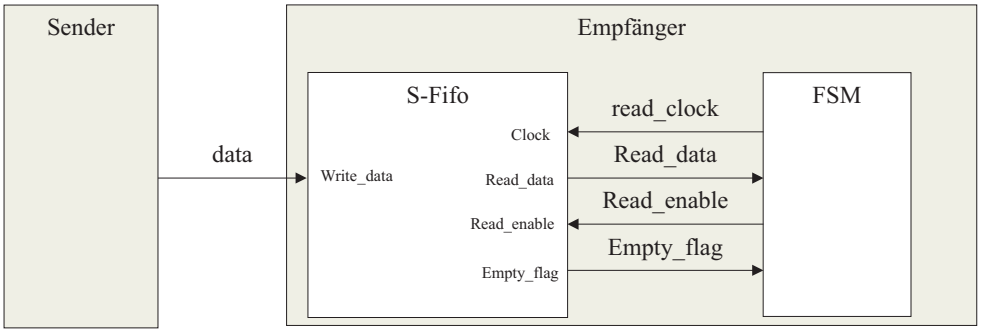


Abbildung 4.15.: Schema einer Channelübertragung [68]

Die Einzelwortübertragung mit Hilfe der SChannels hingegen ist synchron. Da die Taktdomänen des Senders und des Empfängers unterschiedlich sein können, muss zusätzliche Logik den Domänenübergang unterstützen. Eine einfache Methode beruht auf der Verwendung von asynchronen FIFOs und wird in Abb. 4.16 verdeutlicht. Das

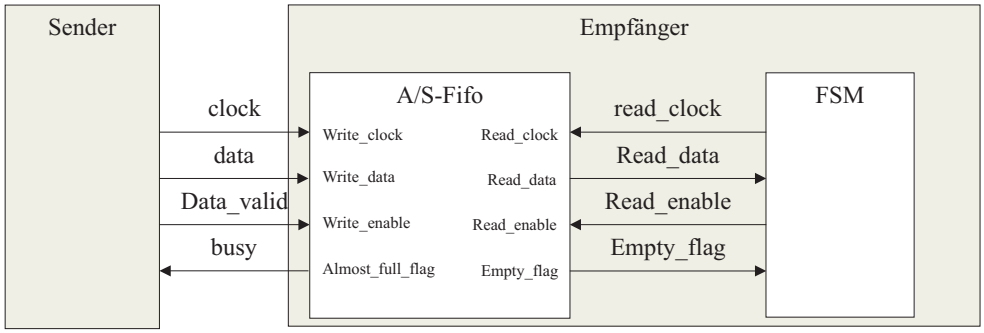


Abbildung 4.16.: Schema einer SChannelübertragung [68]

SChannel-Protokoll sieht vor, dass alle Signale der SChannel-Schnittstelle auf den Takt des Senders bezogen sind. Mit Gültigkeit des Signals `data_valid` wird dem Empfänger signalisiert, dass ein Datenwort mit der nächsten steigenden Taktflanke am Eingang anliegt. Mit Hilfe des Busy-Signals wird zudem garantiert, dass kein Datenwort bei der Übertragung verloren geht. Dazu muss der Sender gewährleisten, dass maximal ein Datenwort nach dem Erkennen von `busy` verschickt wird.

Die dritte Kommunikationsform (Pipe) eignet sich sehr gut für die Mehrwortübertragung von Datenwörtern, die in einem logischen Zusammenhang stehen (Abb. 4.17). Die Zusammenfassung der Datenübertragung in Blöcken ist oftmals aus Sicht der

vom Betriebssystem gekapselten Ressourcen, die zum Beispiel zur Kommunikation zwischen Host-PC und FPGA eingesetzt wird, vorteilhaft. Dies gilt insbesondere für DMA-Transfers über das PCI- bzw. neuere PCIexpress-Protokoll. Zum Beispiel können Bilddaten auf diese Art und Weise effizient zwischen dem Hauptspeicher des Host-PCs und der rekonfigurierbaren Hardware versendet werden. Zusätzlich wird jeder Pipe-

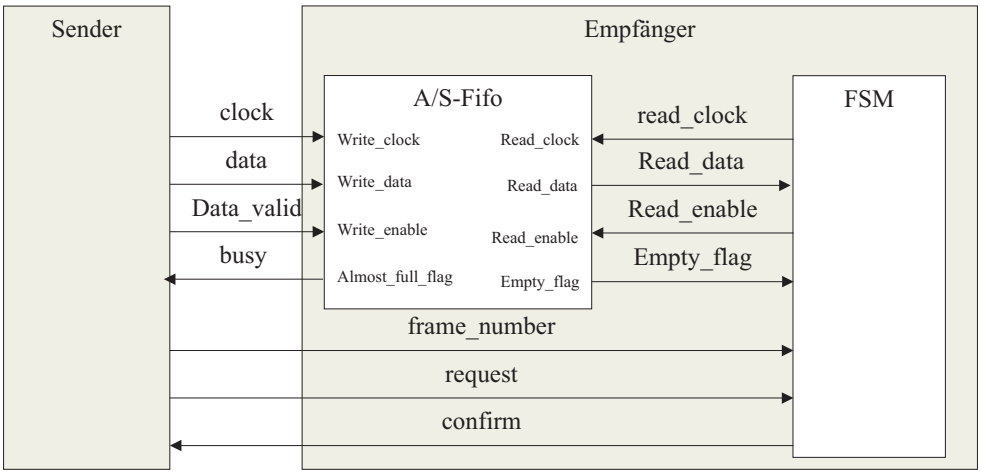


Abbildung 4.17.: Schema der Pipe-Übertragung [68]

Transfer (ein Frame) mit einer Identifikationsnummer versehen. Die Request- und Confirm-Signale dienen dazu, den Start und das Ende eines Frames zu signalisieren.

Im Anschluss an die Transformation durch den Strukturcompiler wird die generierte Quelldatei in einer Simulation bzw. Synthese durch die bekannten Werkzeuge weiterverarbeitet. Eine Beispielkonfiguration für das in dieser Arbeit vorgestellte System zur Gewinnung der Tiefeninformation mit Hilfe konfigurierbarer Hardware ist in section A.1 dargestellt. Dabei handelt es sich um eine Konfiguration für den SGM-Algorithmus auf der Basis verschiedener Hardwareplattformen und Schnittstellen.

5. Stereobildauswertung - Stand der Technik

5.1. Globale Ansätze

Algorithmen, die einen globalen Ansatz verfolgen, sind oftmals sehr komplex, da viele Iterationen, Verbesserungsschritte und Bildsegmentierungen durchgeführt werden müssen. Zudem benötigen sie häufig einen großen Speicher für Zwischenergebnisse. Die globalen Verfahren liefern jedoch die qualitativ besten Ergebnisse.

Die momentan qualitativ besten Verfahren, zu denen beispielsweise Adapting Belief Propagation (AdaptingBP) von A. Klaus et al. [64] gehört, beginnen mit einer Segmentierung des Referenzbildes. Ein einfacher lokaler Matcher errechnet für jedes Pixel innerhalb eines Segments einen passenden Disparitätswert. Anschließend wird eine Ebene in die Menge aller Disparitäten eines Segments eingepasst, die dann in mehreren lokalen Schritten verfeinert wird. Eine globale Optimierung zwischen den einzelnen Segmenten wird am Ende des Verfahrens durch einen Belief-Propagation-Algorithmus (BP) erreicht.

Für Applikationen, bei denen die Laufzeit einen wichtigen Aspekt darstellt, sind diese qualitativ hochwertigen Algorithmen nur bedingt einsetzbar. Das Standard-Belief-Propagation-Verfahren ist zum einen sehr rechenintensiv bezüglich der Wahrscheinlichkeitsverteilungen, die iterativ aktualisiert werden müssen, und zum anderen auch speicherintensiv in der Anwendung, da für jedes Pixel mehrere Verteilungen gespeichert werden müssen. Die meisten BP-basierten Verfahren verwenden aus diesem Grund eine Kombination aus einem hierarchischen [25] und einem asynchronen Ansatz [104] zur Aktualisierung der Verteilungen, wodurch der BP-Algorithmus in weniger Schritten konvergiert.

Das Verfahren RealTimeBP von Q. Yang et al. [118] ist ein Versuch, das hierarchische Belief-Propagation-Verfahren mit Hilfe einer GPU umzusetzen. Durch eine weitere Anpassung der Aktualisierung der Verteilung erreicht RealTimeBP eine Laufzeit von 16 Bilder/Sekunde (B/s) mit 16 Disparitätsstufen und einer QVGA-Auflösung. Das Ergebnis von RealtimeBP ist mit dem von P. F. Felzenszwalb et al. [25] vergleichbar. PlaneFitBP [117] ist mit einem Bild von 512×384 Pixeln pro Sekunde zwar langsamer als RealTimeBP, liefert im Vergleich zu RealTimeBP [118] jedoch ein deutlich besseres Ergebnis. Der Grund für das qualitativ bessere Abschneiden von PlaneFitBP liegt in der zusätzlichen Bildsegmentierung der Eingangsbilder. Beide Varianten nutzen zur

Berechnung handelsübliche Nvidia-Graphikkarten.

Trotz der Zuhilfenahme spezieller Hardware ist aufgrund des nach wie vor hohen Bedarfs an Speicherbandbreite das Auflösungsvermögen beider Verfahren sehr gering. Die Framerate ist für Robotikanwendungen ebenfalls zu niedrig. C.-K. Liang et al. [73] und S. Park et al. [91] reduzieren den Bedarf an Speicherbandbreite zusätzlich durch Kachelung der Daten. Der ASIC-Entwurf von C.-K. Liang et al. [73] erzeugt Disparitätskarten mit einer Auflösung von 640×480 Pixeln und 64 Disparitätsstufen bei einer maximalen Bildwiederholrate von 27 B/s. Sie stellen damit eine Ausnahme dar, da sie, soweit bekannt, als einzige einen anwendungsspezifischen Chip für das Stereo-Matching vorstellten, der auf dem Belief-Propagation-Verfahren beruht. Mit ähnlich großem Hardwareeinsatz, jedoch auf FPGA-Basis, demonstrierten J. M. Perez et al. [93] eine BP-Variante in HD-Auflösung, mit 80 Disparitätsstufen und 25 B/s, indem sie durch Kompression der statistischen Verteilungen Ressourcen einsparten. Die Disparitätskarten von J. M. Perez et al. [93] und C.-K. Liang et al. [73] sind qualitativ ungefähr mit denen vergleichbar, die durch das Verfahren der dynamischen Programmierung (DP) erzeugt wurden, jedoch sind insbesondere zu den Arbeiten von J. M. Perez et al. [93] keine belastbaren Aussagen über die Güte der Disparitätskarte bekannt.

Mit Hilfe der dynamischen Programmierung wird die NP-vollständige Minimierungsaufgabe für das gesamte Bild aus Equation 2.35 in Bildzeilen zerlegt. Der von H. Jeong et al. [60] vorgestellte DP-Algorithmus ist in der Lage, für 1280×1000 Pixel große Bilder und 208 Disparitätsstufen ein Ergebnis in einer Geschwindigkeit von 15 B/s zu liefern. Typisch für den Scanline-Ansatz ist die horizontale Verschmierung der Disparitätskarten, die bei H. Jeong et al. [60] erheblich ist (Abb. 3.1a). Mehrere Verfahren [72, 27, 92] versuchen, benachbarte Bildzeilen während der Kostenbildung oder der Nachbearbeitung zu integrieren. Die Ergebnisse von dynamisch programmierten Matchern bleiben jedoch denen der BP-Matcher unterlegen. Der Ansatz von S. Mattoccia et al. [82] erzielt für einen DP-Ansatz sehr gute Ergebnisse, ist jedoch aufgrund der notwendigen Farbsegmentierung nicht mehr echtzeitfähig.

Verfahren	∅ Fehler	Rang	B/s	Bildgröße	Disp	Hardware
PlaneFitBP [117] △	5,78	11	1	512×384	48	GPU
RealTimeBFV [63] ∇	7,65	44,2	12	450×375	64	GPU
RealTimeBP [118] △	7,69	43,6	16	320×240	16	GPU
RealtimeABW [15]∇	7,9	40,1	2	384×288	15	CPU
FastAggreg [107] ∇	8,42	48,4	5	384×288	16	CPU
RealTimeVar [66]△	9,05	51,2	2	384×288	16	CPU
RealTimeCensus [58] ∇	9,73	56,1	573	320×240	15	GPU
RealTimeGPU [115]♢	9,82	56,9	43	320×240	16	GPU
ReliabilityDP [85] ♢	10,7	60,2	25	512×512	16	GPU

Tabelle 5.1.: Bewertung echtzeitfähiger Stereo-Matching-Verfahren nach [97]. Legende: ∇=lokales, △=globales und ♢ = hybrides Verfahren (Fehler > 1 Pixel)

5.2. Lokale Verfahren

Bereits in den frühen 90er Jahren sind für Echtzeitanwendungen auch lokale Ansätze verwendet worden (Table 5.2). Trotz der relativ einfachen Berechnung der üblicherweise verwendeten SAD-, SSD- und NCC-Kostenmaße[1] musste aufgrund der zur damaligen Zeit sehr beschränkten Kapazität der PC-Systeme auf sehr komplexe Spezialhardware zurückgegriffen werden. O. Faugeras et al. [24] stellten 1993 einen vielbeachteten Kreuzkorrelator als Matcher vor. Eine von DEC entwickelte FPGA-Matrix (PerLe-1) mit 16 FPGAs der ersten Generation vom Typ Xilinx XC3090 konnte mit ca. 3 B/s Tiefenkarten mit einer Auflösung von $255 \times 256 \times 32$ Pixel berechnen.

Einen ähnlich großen Hardwareaufbau verwenden J. Woodfill et al. [116]. Deren FPGA-Matrix mit ebenfalls 16 FPGAs vom Typ Xilinx XC4000 errechnet auf der Grundlage der von R. Zabih et al. [120] vorgeschlagenen Census-Transformation Tiefenkarten mit 32 Disparitätsstufen für QVGA-Bilder in 21 ms (46 Hz).

Die am häufigsten verwendete Kostenfunktion lokaler Verfahren basiert auf dem SSD- bzw. SAD-Kostenmaß zum Vergleich zweier Grauwerte. Weil die pixelbasierten Kostenfunktionen sehr anfällig gegenüber Rauschen sind, werden zur Steigerung des Signal-Rausch-Verhältnisses die pixelbasierten Kosten über eine lokale Umgebung aufsummiert. Ein Nachteil in der Verwendung dieser Fensterung besteht darin, dass die Disparitätsübergänge nicht ausreichend differenziert werden können. Abhilfe schaffen hier dynamische Fensterfunktionen, die die Qualität der Disparitätskarten weiter verbessern. Eine aktuelle Übersicht der verschiedenen Fensterfunktionen geben F. Tombari et al. [109] und M. Gong et al. [37].

Zur Verbesserung der Qualität der Tiefenkarte verfolgen M. Kuhn et al. [69] einen hybriden Ansatz, in dem sie die Ergebnisse aus SSD und Census kombinieren und zugleich in einem ASIC implementieren. Das Auflösungsvermögen ihrer ASICs ist jedoch relativ gering und beträgt lediglich 256×192 Pixel bei 25 Disparitätsstufen und 50 B/s.

Mehr Auflösung bieten die von Y. Miyajima et al. [86] und D. Han et al. [41] vorgestellten FPGA-basierten Systeme. Ersteres ist in der Lage, VGA-Bilder mit 20 B/s zu berechnen. Hervorzuheben ist die mögliche Anzahl der Disparitätsstufen von maximal 200 Pixeln. Mehrere Versuche sind unternommen worden, durch Filterstufen in Vorverarbeitungsprozessen die Anfälligkeit der lokalen Matcher gegenüber Rauschen zu minimieren. Standard-Gaußfilter oder Mittelwertfilter haben einen schlechten Einfluss auf die Kantentreue, der sich in den Tiefenkarten widerspiegelt. S.-K. Han et al. [42] verwenden aus diesem Grund einen relativ einfachen adaptiven bilateralen Filter von C. Tomasi et al. [106], der kantenerhaltend wirkt.

Einen Stereo-Matcher, bei dem die Bildwiederholrate im Vordergrund steht, zeigen K. Ambrosch et al. [5]. Deren lokaler Matcher nutzt das SAD-Maß und ist in der Lage, 450×375 Pixel große Bilder mit 150 Disparitätsstufen mit 600 B/s zu berechnen.

[1] *normalized cross-correlation*

Verfahren	Methode	B/s	Bildgröße	max. Disparität	Hardware
Faugras [24]	CC [2]	≈ 7	256 × 256	32	16 FPGAs
Woodfill [116]	Census	42	320 × 240	24	16 FPGAs
Kuhn [69]	Census	50	256 × 192	25	ASIC
Miyajima [86]	SAD	80	320 × 240	200	4 FPGAs
Ambrosch [67]	SAD	425	320 × 240	100	FPGA
Masrani [80]	LW-Phase-Correlation	30	480 × 640	128	FPGA

Tabelle 5.2.: Eine Auswahl lokaler echtzeitfähiger Stereo-Matcher und deren Realisierung

Auf Grund der hohen Hardwarekosten sind aber nur die wenigsten echtzeitfähigen Systeme für den produktiven Einsatz geeignet. C. Murphy et al. [88] demonstrieren auf der Grundlage eines Low-Cost-FPGAs vom Typ Xilinx Spartan 3 einen Census-basierten Matcher, der 150 QVGA-Bilder pro Sekunde mit max. 20 Disparitätsstufen liefert.

Zusätzlich seien Filter-Ansätze erwähnt. Sowohl das Referenzbild als auch das zu matchende Bild werden dabei in Vorverarbeitungsschritten mit zum Beispiel einem Gabor-Filter gefaltet. Die Phasenverschiebung beider Filterantworten lässt Rückschlüsse auf die Disparität zu. Die Amplitude kann als Konfidenzmaß verwendet werden. D. K. Masrani et al. [80] demonstrieren ein System auf der Grundlage des Local-Weighted-Phase-Correlation-Algorithmus. Das System berechnet 128 Disparitätsstufen für VGA-Bilder in 30 B/s. Um dieses Ziel zu erreichen, wird jedoch eine FPGA-Matrix mit 4 Altera Stratix FPGAs benötigt.

Neben FPGAs, DSPs und ASICs sind in den letzten Jahren verstärkt GPU-Implementierungen gezeigt worden. Graphikkarten werden in sehr großen Stückzahlen vor allem für den Verbrauchermarkt produziert und sind deshalb relativ kostengünstig. Die von ihrem Graphikprozessor, der GPU, zur Verfügung gestellten parallelen Strukturen und die relativ hohe Speicherbandbreite der Graphikkarte machen die GPU-Lösung attraktiv für die Bildverarbeitung. Insbesondere die Programmierbarkeit und damit bessere Zugänglichkeit der GPU-Hardware ist für Wissenschaftler, die mit der Softwareentwicklung vertrauter sind, ein Vorteil gegenüber dem klassischen Hardwareentwurf. Zudem bieten die GPU-Hersteller modifizierte Varianten für das wissenschaftliche Umfeld an, mit denen exakte Rechenoperationen möglich sind.

[2] Verschiedene Varianten der Kreuzkorrelation

5.3. Semi-globale Verfahren

Hybride Verfahren kombinieren die globalen Minimierungsansätze mit lokalen Methoden in der Absicht, durch Kombination beider Strategien gute Ergebnisse und reduzierte Komplexität zu gewährleisten (Table 5.3). I. Ernst et al. [22] präsentieren eine GPU-Implementierung des Semi-Global-Matching-Verfahrens (SGM), das auf dem von H. Hirschmüller [50] vorgestellten Algorithmus basiert, der in Equation 6.2 detailliert vorgestellt wird. Die verwendete Graphikkarte berechnet Tiefenbilder in VGA-Auflösung mit ca. 4 B/s.

Die sehr guten Ergebnisse bei der Zuordnung der Korrespondenzen und die relativ geringe Komplexität des SGM-Verfahrens führten dazu, dass der Algorithmus in unterschiedlichen Ansätzen weiterentwickelt worden ist. Das Ziel besteht vorrangig darin, den Algorithmus für Echtzeit-Anwendungen nutzbar zu machen. S. K. Gehrig et al. [31] präsentieren eine FPGA-Implementierung, mit deren Hilfe Bilder der Größe 340 × 200 Pixel verarbeitet werden können. Die maximale Anzahl an Disparitätsstufen beträgt 64. In T. Vaudrey et al. [112] werden auf der Grundlage des SGM-Algorithmus Aufnahmen unter Laborbedingungen mit realen Daten verglichen. Ähnliche Betrachtungen für den Bereich der Fernerkundung wurden jeweils von M. Heinrichs et al. [45], S. Gehrke et al. [33] und P. d'Angelo et al. [18] durchgeführt.

Durch die Aufnahme weiterer Parameter zu den ursprünglich vorhandenen Strafkosten P_1 und P_2 erhofften sich S. Hermann et al. [47] das Matching von SGM zu verbessern. Das hatte zur Folge, dass die Matching-Ergebnisse stark von den gewählten Parametern abhingen, und damit ein weiterer Vorteil von SGM, nämlich die relativ große Toleranz gegenüber einer Vielzahl realistischer Szenen, aufgegeben werden musste.

M. Humenberger et al. [57] untersuchen in ihrer Arbeit die Kombination des SGM-Algorithmus mit verschiedenen Kostenfunktionen und Segmentierungsschritten zur Verfeinerung der Disparitätenkarte. In H. Hirschmüller et al. [55] werden dieselben Betrachtungen detaillierter durchgeführt sowie in H. Hirschmüller [48] eine Variante des SGM-Algorithmus vorgestellt, ebenfalls mit Bildsegmentierungsstufen zur Verbesserung der Disparität.

Eine weitere FPGA-Implementierung des SGM-Verfahrens stammt von C. Banz et al. [7], deren Ansatz die Berechnung der Kosten nicht nur parallel bezüglich der Suchrichtung sondern auch für mehrere Zeilen gleichzeitig durchführt und dadurch die notwendige Bandbreite zum Zwischenspeicher reduziert. Die hohen Datenraten der gezeigten FPGA-Realisierung wurden durch Weglassen eines wesentlichen Teilschrittes erreicht, was zu einem Qualitätsverlust der Disparitätskarte führte.

Aufgrund der Komplexität der Entwicklung von Hardwarelösungen, insbesondere jener, die auf rekonfigurierbaren Logikbausteinen basieren, stellen S. K. Gehrig et al. [32] eine CPU-Implementierung des SGM-Algorithmus vor, die in QVGA-Auflösung und 64 Disparitätsstufen etwas mehr als 12 Tiefenkarten pro Sekunde erzeugt.

[3] *hierarchical mutual information*

Verfahren	Kostenmaß	B/s	Bildgröße	maximale Disparität	Plattform
GPU-SGM [22]	HMI[3]	13	320 × 240	128	GPU
CPU-SGM [32]	Census	≈ 13	320 × 240	64 (sampled)	Intel 975ex
Daimler-SGM [31]	SAD	25	320 × 200	64 (sampled)	FPGA

Tabelle 5.3.: SGM-Realisierungen

Der Vollständigkeit halber sei auf die Arbeit von B. McCullagh [83] hingewiesen, die einen Cell-B/E-Chip[4]als Hardwarebasis verwenden und dafür einen lokalen Matcher basierend auf dem SAD-Kostenmaß implementiert haben.

5.4. Kommerzielle Systeme

Historisch betrachtet entwickelten sich die ersten rechnergestützten Stereo-Vision-Systeme aus dem Bereich der Photogrammetrie zur Erstellung von Höhenkarten. Anfänglich brachte noch ein Bediener überlappende Bildbereiche zueinander in Bezug, dann löste die automatische Korrespondenzanalyse diesen ab und beschleunigte damit die Verarbeitung erheblich. Die Einführung von digitalen optischen Zeilen- und Matrixsensoren unterstützte diesen Entwicklungsgang zusätzlich.

Die überwiegende Zahl kommerzieller, auf dem freien Markt erhältlicher, Stereo-Vision-Systeme legt den Schwerpunkt auf die Echtzeitfähigkeit der Datenverarbeitung (Table 5.4). Die Hersteller argumentieren relativ häufig mit der Notwendigkeit, mit der hohen Geschwindigkeit automatisierter Montagestraßen schritthalten zu müssen, um für ihren vorwiegend industriellen Kundenkreis interessant zu sein. Damit unterscheiden sich die Anforderungen an das Stereo-Vision-System erheblich von denen im akademischen Umfeld. Ohne zu sehr verallgemeinern zu wollen, spielen in der Industrie die Faktoren Laufzeit, Größe, Kosten und Energieverbrauch eine übergeordnete Rolle, und Flexibilität und Qualität müssen sich dem unterordnen. So ist auch zu erklären, warum nur einfache Stereo-Matching-Verfahren realisiert wurden. Die Möglichkeit, ein kommerzielles System als Grundlage für Forschung und Entwicklung zu nutzen, soll hier aber nicht ausgeschlossen werden.

Diverse Firmen bieten FPGA-basierte Stereo-Vision-Systeme an, die vorrangig für die industrielle Prozessautomatisierung bestimmt sind. Die Framerate, mit der die Disparitätenkarte geliefert wird, liegt gewöhnlich bei 25–30 Hz. Die gelieferte Auflösung des Sensors beträgt 640 × 480 Pixel und entspricht der Standard-VGA-Auflösung. Das von Videre Design LLC [19] vertriebene Stereo-Vision-System STOC hat eine Tiefenauflösung von 64 Disparitätsstufen. Die Eingangsdaten werden mit einem Faktor 16 unterabgetastet. Als FPGA kommt ein kostengünstiger Xilinx Spartan3-1000 zum Einsatz. Das nDepth-Vision-System von FocusRobotics Ltd. verfolgt ein ähnliches Konzept wie auch die ersten Systeme von Valde Systems. Getrennt vom

[4]Cell Broadband Engine von Sony, IBM und Toshiba

Hersteller	max. Bild größe in Pixel	max. Bildrate	max. Disparität	Kostenmaß	Plattform
Videre Design STOC	640 × 480 (VGA)	30	64	SAD	DSP / FPGA
FocusRobotics nDepth-Vision	752 × 480 (WVGA)	30	92	SAD	PCI-FPGA-Karte
Tyzx DeepSea 2	512 × 2048	30	52	Census	Vollkunden-IC
Valde Systems VS1701	640 × 480 (VGA)	offenes System [5]			DSP / FPGA

Tabelle 5.4.: Übersicht kommerzieller Stereo-Vision-Systeme

Kamerakopf werden die Stereobilder wie im Falle des nDepth-Vision-Systems [95] von Focus Robotics PC-seitig auf einer Vollkunden-IC-Karte berechnet. Das Ergebnis wird anschließend im Speicher des Host-PCs abgelegt. Valde Systems ist an dieser Stelle erheblich flexibler und erlaubt durch die Verwendung eines DSPs eine nachträgliche Programmierung des Stereo-Matching-Cores.

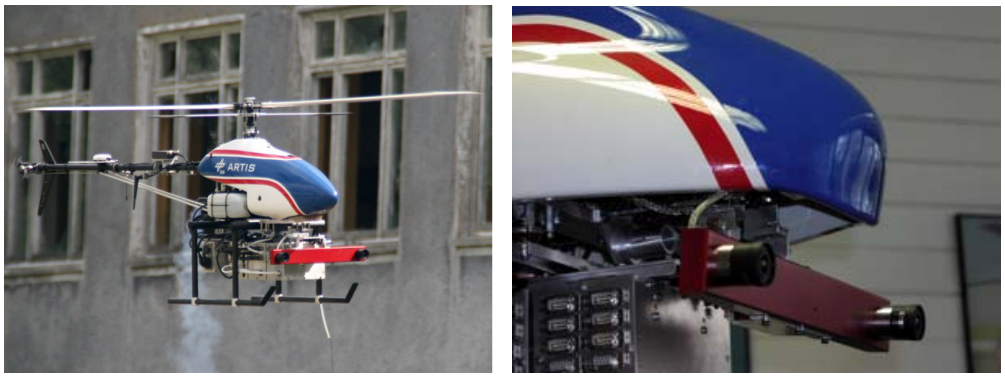
Valde Systems VS1701 [111] (Abb. 5.1) ist ein speziell für den Außenbereich entworfenes Kamerasystem, das optional mit einem für Anwenderlogik freien FPGA (Altera Stratix II) erhältlich ist. Der FPGA kann sowohl für Bildvorverarbeitungsaufgaben als auch für Stereo-Matching-Aufgaben verwendet werden. Der Hauptteil der Bildverarbeitung wird jedoch von einem DSP übernommen, der in einer PC-ähnlichen Umgebung eingebettet ist.

Abbildung 5.1.: Ein Stereokamerasystem von Valde Systems VS1701 [111]

Die Systeme von Minoru 3D, Point Grey Research Inc. und Nvela sind einfache Stereo-Kamera-Köpfe, die ohne zusätzliche Datenverarbeitungseinheiten vertrieben werden und die mehr als nur den Bildeinzug vornehmen. Unter realen Einsatzbedingungen

[5] Valde Systems liefert keinen eigenen Stereo-Matching-Core. Es existieren jedoch mind. zwei Stereo-Matching-Systeme, die entweder die im Kamerakopf verbauten DSPs und FPGAs nutzen [110] oder den mit Firewire angeschlossenen Imageprozessor (DSP) [105].

zeigen die kommerziellen Systeme, die ursprünglich für den mobilen Einsatz gedacht sind, ihre Schwächen. Insbesondere bei schlechten Signal-Rausch-Verhältnissen liefern die verwendeten Stereo-Matcher unbefriedigende Ergebnisse.



(a) Artis UAV neben einem verlassen Gebäudekomplex (Rosenkrug)

(b) Nahaufnahme des Systems Videre Design STOC [19] an einem Versuchsträger

Abbildung 5.2.: Artis UAV (DLR-Institut für Flugsystemtechnik)

Das DLR-Institut für Flugsystemtechnik erprobt unter anderem UAVs, die das kommerzielle Stereo-Matching-System STOC von Videre Design [19] verwenden. Der Stereo-Kameraaufbau mit fester Basislänge wird an die Spitze des Gerätes montiert (Abb. 5.2a).

(a) Rosenkrug-Aufnahme linke Kamera in VGA-Auflösung

(b) Disparitätskarte der Rosenkrug-Aufnahme

Abbildung 5.3.: Rosenkrug (DLR-Institut für Flugsystemtechnik)

In Abb. 5.3a und 5.4a sind Momentaufnahmen aus zwei unterschiedlichen Szenarien zu erkennen. Dabei handelt es sich bei der ersten Aufnahme um ein verlassenes Kasernengelände. Die zweite Messreihe wurde auf einem präparierten Versuchsfeld durchgeführt. Jeweils rechts der Eingangsbilder sind die vom lokalen Matcher erstellten Disparitätsbilder zu erkennen. Insbesondere die Aufnahmen des Kasernengeländes

(a) Versuchsfeld mit künstlichen Markierungen (b) Disparitätskarte des Versuchsfeldes

Abbildung 5.4.: Versuchsfeld mit Zielmarkierungen zur Validierung der Algorithmen (DLR-Institut für Flugsystemtechnik)

machen die ungenügende Qualität des Matchers deutlich, und zwar durch die vielen nicht korrekt zugeordneten Teilflächen des Bodens und der Häuserwand.

Bewertung der Algorithmen

Ein etablierter Datensatz zur Bewertung von Stereo-Matching-Algorithmen, die dichte Tiefenbilder erzeugen, stammt von D. Scharstein et al. [97]. Die Qualität der Disparitätenbilder wird durch Aufsummierung der Fehlzuweisung bestimmt. Als Vergleichsgrundlage dient eine vorher vermessene Referenz. Abbildungen der Datensätze befinden sich im Anhang section A.3.

6. Entwurf eines echtzeitfähigen Systems

In diesem Kapitel wird ein Entwurf für ein echtzeitfähiges System erarbeitet, mit dessen Hilfe Tiefeninformationen aus Stereobildern gewonnen werden können. Der Entwurf berücksichtigt die für eine Realisierung nötige bzw. zur Verfügung stehende Hardware, die Algorithmen und die Sensorik. Ausgangspunkt ist die Ableitung der Anforderungen aus der konkreten Anwendung, denn jeder Entwurf ist Teil einer Menge von möglichen Entwurfsvarianten. Diese Menge wird durch weiche und harte Bedingungen begrenzt.

6.1. Anforderungsanalyse

Robotikapplikationen stehen im Vordergrund dieser Arbeit, so dass auf dieses Anwendungsszenario einleitend besonders eingegangen wird. Wesentliche Kriterien zur Bewertung des Entwurfs unter automatisierungstechnischen Aspekten und stereoskopischen Auswertungen sind der gewünschte Messbereich und in diesem Zusammenhang die Genauigkeit der Messung, die Zuverlässigkeit des Systems und das Zeitverhalten während der Berechnung.

Sowohl der Messbereich als auch die Genauigkeit der registrierten physikalischen Größen haben Einfluss auf die Güte der abgeleiteten Information. Im Falle von Navigationsanwendungen sind das zum Beispiel Lage und Geschwindigkeit.

Eine Aussage über die Zuverlässigkeit des Gesamtsystems kann über statistische Kenngrößen des zu erwartenden Fehlverhaltens gewonnen werden. Eine Fehlfunktion ist dabei eine vom Anwender beobachtete Abweichung von einer erwarteten Reaktion des Systems. Das System geht dabei in einen nicht beabsichtigten Fehlerzustand über. Dieser Übergang wird durch eine Fehlerquelle ausgelöst. Die Zuverlässigkeit lässt sich empirisch oder analytisch schätzen.

Das Zeitverhalten eines Systems besagt, inwiefern und gegebenenfalls in welchem Zeitraum eine Reaktion des Systems zu erwarten ist. Von einem Echtzeitverhalten wird gesprochen, wenn die Systemantwort innerhalb einer definierten Zeit bereitsteht. Genauer kann zwischen harter und weicher Echtzeit gesprochen werden. Weiche Echtzeit bedeutet, dass lediglich für den Erwartungswert eine obere Schranke angegeben werden kann. Im Sinne einer harten Echtzeit befindet sich jede Reaktionsdauer innerhalb einer Obergrenze.

Aufgrund der Vielzahl unterschiedlichster Hardwarekomponenten, Sensoren und Stereo-Matching-Algorithmen die zur Verfügung steht, ist ein strukturiertes Vorge-

hen notwendig. Die Methoden des HW/SW-Entwurfs müssen den Integrationsprozess unterstützen, mit dem Ziel, eine Integration zu erarbeiten, die die gegensätzlichen Parameter des Gesamtsystems wie zum Beispiel Beschaffungskosten, Geschwindigkeit, physikalische Größe und Energieverbrauch berücksichtigt.

Robotikanwendungen

Je nach Einsatzort und Aufgabenstellung ergeben sich unterschiedliche Anforderungen, die im Folgenden dargelegt werden. Prinzipiell lassen sich Robotiksysteme in zwei große Teilkategorien bzgl. ihrer Mobilität einordnen. Stationäre Systeme werden vorwiegend in voll automatisierten Industriestraßen eingesetzt. Einzelne Manipulatoren besitzen einen fest definierten Aktionsraum, den sie mit ihrer Mechanik maximal erreichen können. In diesen, auch Arbeitszellen genannten Bereichen, entsteht eine Dynamik, indem zum Beispiel Werkstücke von außen in die Zelle eingebracht werden. Dem Roboter steht nur im begrenzten Umfang Vorwissen bezüglich der Lage des Werkstoffes zur Verfügung. Eine denkbare Aufgabe des Sensorsystems ist es daher, die Lage zu erfassen, wenn sich prozessbedingt die Werkstücke nicht an vordefinierten Stellen befinden. Durch Lokalisierung des Objekts und Erfassung der Eigenbewegung des Roboterarms wird der Roboter in die Lage versetzt, das Objekt zu greifen und in irgendeiner Form zu manipulieren.

Eine Vielzahl an Szenarien ist vorstellbar, in denen die Ausgangssituation erschwert wird. In abgeschlossenen Zellen wird davon ausgegangen, dass der Arm ein genaues Modell der Werkstücke besitzt und nur diese auch in seinem Bereich eintreten. Bei unvorhersagbaren Ereignissen, die nicht in die Modellwelt des Roboters passen, schaltet dieser sich ab, bis der Mensch den Arbeitsraum wieder freigibt. Diese Situation kommt in Industrieanlagen häufig vor und verzögert den Produktionsprozess. Idealerweise sind die Arbeitsbereiche der autonom agierenden Roboter und der menschlichen Arbeiter miteinander verschmolzen. Das bedeutet, dass die Zelle nicht mehr streng von der Außenwelt abgetrennt ist und der Roboter auf ihn unbekannte Objekte adäquat reagieren muss. Es ist wünschenswert, dass sich Menschen zwischen den Robotern frei bewegen können, ohne dass diese eine Gefahr darstellen, und beide Parteien ihre jeweiligen Arbeiten parallel weiterführen. Um diese relativ schwierige Aufgabe zu lösen, muss der Roboter sein Umfeld und seine Situation in diesem Umfeld wahrnehmen. In diesem Anwendungsfall ist eine Aufgabe des Sensorik-Subsystems, Hindernisse bzw. einzelne Objekte zu lokalisieren. Kennt der Roboter seine eigene Lage, wird im Fall der Lokalisierung und Verfolgung weiterer Objekte auch von Tracking gesprochen.

Mobile Roboter sind hingegen üblicherweise mit der Situation konfrontiert, keine Kenntnis darüber zu besitzen, wo und wie sie sich eingangs befinden. Sie starten in Umgebungen, von denen sie a priori keine Information besitzen und verfügen über keine Daten hinsichtlich ihrer eigenen Lage. Zusätzlich kann sich ihre Lage durch Krafteinwirkung von außen unvorhergesehen ändern. Die bereits lokal registrierte Trajektorie bzw. lokale Karte wird ungültig und eine neue Karte muss erstellt werden. Die Aufgabe, gleichzeitig die Umgebung und die eigene Lage in ihr zu ermitteln, wird

allgemein unter den Begriff SLAM (*simultaneous localization and mapping*) geführt. Lokalisierungs- und Mappingaufgaben sind eng miteinander verknüpft. SLAM stellt somit eine Erweiterung des Trackings von einem Objekt (sich selbst) auf viele Objekte (die Umwelt) dar. Der am häufigsten verwendete Ansatz zur Lösung von SLAM ist die Verwendung von erweiterten Kalmanfiltern zur Fusion mehrerer Sensoren. Durch Fusion von Daten aus Odometrie, GPS, Inertialsystemen und Entfernungssensoren kann der Roboter seine Umgebung erforschen und in ihr navigieren.

Lokalisation

Als visuelle Odometrie wird das Messverfahren bezeichnet, mit dessen Hilfe Daten aus optischen Sensoren zur Unterstützung der Navigation gewonnen werden. Häufig werden Feature-Tracker verwendet, die in zeitlich versetzten Aufnahmen Landmarken wieder erkennen. Aus der Verschiebung der Features im Bild kann auf die relative Bewegung im Objektraum geschlossen werden. Typische Features sind Kanten und Ecken im Raum, die von den Sensoren erfasst werden. Das Hauptproblem besteht darin, die korrespondierenden Features einander zuverlässig zuzuordnen. Zur Anwendung dieses Verfahrens ist im Prinzip nur ein optischer Sensor notwendig. Allerdings sind binokulare Systeme nur unwesentlich teurer im Aufbau als Monosysteme, wohingegen die Tracking-Algorithmen mit ihnen günstigere Voraussetzungen gegenüber Verdeckungen und Fehlerintegration vorfinden.

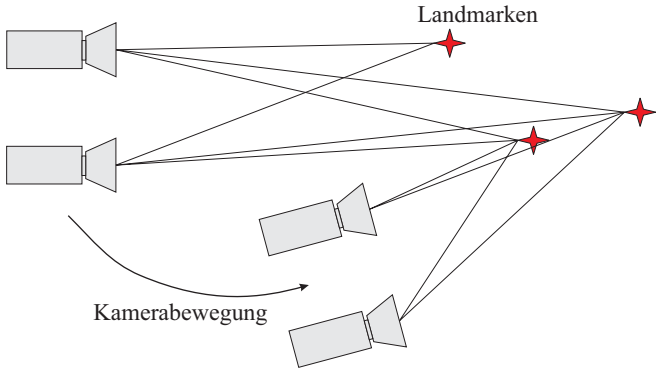


Abbildung 6.1.: Gleichzeitige Triangulation und Ego-Motion-Estimation

Für die unterschiedlichen Aufbauten können die bekannten Konzepte aus der Übersicht in Abb. 1.1 übernommen werden. Die Abb. 6.1 verdeutlicht die gleichzeitige Triangulation von Landmarken und Schätzung der Eigenbewegung anhand von Stereoaufnahmen. Sind die Parameter der Orientierungen beider Kameras bekannt, kann im Gegensatz zum monokularen Fall die absolute Eigenbewegung ermittelt werden. Häufig werden Inertialsysteme und Weggeber der Mechanik verwendet, weil mit ihrer Hilfe der Suchbereich für die Korrespondenzenanalyse eingegrenzt werden kann.

Bisherige Arbeiten zu echtzeitfähigen Navigationslösungen beschränken sich auf Sparse-Stereo-Matcher. Aus Effizienzgründen wird die Korrespondenzanalyse typi-

scherweise sowohl zwischen zeitlichen Bildpaaren als auch örtlichen Stereosets mit einfachen lokalen Matchern wie zum Beispiel SAD realisiert. Dense-Stereo-Matching hingegen kann zum einen die Featureauswahl vereinfachen, zum anderen können Eigenbewegungen hinsichtlich Translation und Rotation separiert werden. Zum Beispiel wird für Pixel mit einer Disparität gleich Null ausschließlich eine Rotationsbewegung angenommen. Im Vordergrund für Lokalisierungsanwendungen stehen hohe Abtastraten und ein weiter Öffnungswinkel bei geringem Auflösungsvermögen.

Mapping/Exploration

Für Mapping- und Explorationsaufgaben sind ein großer Sichtbereich, eine schnelle Abtastrate und ein großes Auflösungsvermögen wünschenswert, da im Zentrum der Exploration häufig die Hinderniserkennung und die Erforschung der eigenen Umgehung stehen. Eine Vielzahl an Sensoren steht zur Verfügung, um Objekte auf Distanz zu erkennen. Laser-Scanner haben eine sehr hohe Messgenauigkeit und bieten den Vorteil, technologiebedingt bereits eine Tiefeninformation zu liefern. Radarsysteme messen ebenfalls auf großen Entfernungen sehr genau und sind zudem kostengünstig verfügbar. Kamerasysteme bieten eine Kombination aus einem relativ großen FoV, einer sehr hohen Abtastrate und einem sehr großen Auflösungsvermögen. Insbesondere die Abtastrate ist für die Objekterkennung von Bedeutung, denn zu geringe Abtastraten wirken sich nachteilig in Anwendungen aus, bei denen Objekte verfolgt bzw. erkannt werden müssen, die sich schnell bewegen.

Objektwiedererkennung

In ähnlicher Art und Weise wie für die Objekterkennung ist es auch hier sinnvoll, einen Sensoraufbau zu verwenden, der einen weiten Öffnungswinkel bzw. FoV aufweist. Im Mittelpunkt stehen Merkmale an den Objekten, die es ermöglichen, diese eindeutig in Folgeaufnahmen zu identifizieren. Ein einfacher Sparse-Stereo-Matcher kann verwendet werden, um nach den Merkmalen für die Objekte in den Aufnahmen suchen. Eine übergeordnete Rolle spielt die Anforderung an die Berechnungsdauer.

Oberflächenmodellierung

Für Modellierungsaufgaben im messtechnisch relevanten Bereich sind sehr hohe geometrische Auflösungen mit einem sehr guten SNR gefordert. Typische Szenarien sind 3-D-Modellierung von Freiformen für die Qualitätssicherung im Ingenieurwesen. Messūngenauigkeiten bzw. verdeckte Bereiche müssen durch Neuausrichtung der Kamera kompensiert werden bzw. können eventuell in Nachverarbeitungssschritten durch Filter ausgeglichen werden.

Indoor-Einsatz

Im Innenbereich können vertikale und horizontale Strukturen von Fluren und Raumwänden als vorgegebene Information eingebunden werden. Ebenso wirkt sich die Größe

der Räumlichkeiten auf das Sensorkonzept bezüglich des gewählten Sichtbereichs der Sensoren und deren Anordnung aus. Durch die Hinzunahme aktiver Lichtquellen kann der Nahbereich der passiven Sensoren zusätzlich ausgeleuchtet werden.

Outdoor-Einsatz

Im Vergleich zum Innenbereich, lassen sich die Beleuchtungsverhältnisse in der freien Umgebung nur sehr schwer bzw. überhaupt nicht kontrollieren, so dass passive optische Verfahren variable Aufnahmebedingungen vorfinden, die nur begrenzt von vornherein kompensiert werden können. Schnelle Rotationsbewegungen führen dazu, dass bereits gefundene Merkmale in großer Entfernung schnell aus dem Sichtbereich des Sensors verschwinden. Für Kameras, die zum Verfolgen von Merkmalen eingesetzt werden, verschärft sich dieses Problem in ländlichen Gebieten, für die es generell schwieriger ist, robuste Features zu finden. Mögliche Witterungseinflüsse wie Nebel, Schnee, Staub, Regen, Gegenlicht etc. müssen bei der Sensorauswahl mit in Betracht gezogen werden.

Allgemeine Betrachtungen bezüglich des Gesamtsystems betreffen den Energiehaushalt und die Größe des Systems, die für mobile Systeme verstärkt in den Vordergrund treten.

Anforderungen an die Mechanik

Abhängig vom Einsatzgebiet und den zu erwartenden äußeren Einflüssen auf die Mechanik, müssen die Merkmale Temperatur, elektromagnetische Verträglichkeit, Vibrationen, Shockfestigkeit, Feuchtigkeit in einem Gesamtsystem mit berücksichtigt werden. Feuchtigkeit auf den Detektoren kann gravierende Nebeneffekte bis hin zur Zerstören der Sensoren bedeuten.

Ein wesentlicher Aspekt der Mechanik ist auch die Dauerhaftigkeit der Kalibrierung. Insbesondere in solchen Multisensorsystemen, in denen die Sensoren relativ weit entfernt voneinander montiert sind, ist der kalibrierte Zustand nur mit hohem Aufwand zu gewährleisten. Zudem sind die mechanischen Verbindungen der Optik bereits bei moderaten Belastungen anfällig für Veränderungen, die den kalibrierten Zustand der Messeinrichtung zerstören. Des Weiteren ist ein verlässlicher operationeller Betrieb nur mit sicheren elektro-mechanischen Verbindungen möglich. Unterbrechungen von Signal- oder Energieleitungen dürfen nicht vorkommen bzw. müssen auf ein absolutes Minimum reduziert werden, da sich Messungen über Stunden erstrecken können.

Luft- und Satellitenbildauswertung

Für die Erstellung von 3D-Oberflächen- und Höhenmodellen im Bereich der Luft- und Satellitenfernerkundung werden sehr große Bildverbände erwartet. Die erfasste Parallaxe kann ebenfalls beträchtlich sein. Insbesondere für Aufnahmen von Gebirgen kann der Suchbereich für das Stereo-Matching sehr groß werden.

Luftbildaufnahmen mit Zeilenkameras die zeilenweise das Gebiet abtasten, können

nicht ohne weiteres vollständig rektifiziert werden, sodass keine geradlinige Epipolargeometrie angenommen werden kann [56]. Approximativ kann die Epipolarlinie aber stückweise linearisiert werden. Anschließend werden innerhalb eines kleines Suchfensters um diese Approximation herum die korrespondierenden Punktpaare gesucht. Alternativ können die komplexen Epipolargeometrien durch ein genaues Höhenmodell berechnet werden.

Im Gegensatz zur Nahbereichsphotogrammetrie kommen in Luft- und Satellitenaufnahmen untexturierte Bereiche seltener bzw. häufig nur in Form von Wolken, Wasser und Schnee vor. Durch Ausmaskierungen in Vorverarbeitungsschritten können diese Anteile reduziert werden.

6.2. Systemkonzept

Nachdem die Rahmenbedingungen und Einsatzszenarien in der Robotik und der Geofernerkundung skizziert worden sind, soll in dem folgenden Teil auf den Entwurf eines Systems zur Generierung von Tiefeninformation aus Stereobildern eingegangen werden.

Ein übergreifender Ansatz für ein echtzeitfähiges System zur Rekonstruktion der Tiefeninformation aus Stereoaufnahmen muss sowohl die Komplexität und die Qualität der Algorithmen als auch die Vielfalt unterschiedlicher Hardwareplattformen in Betracht ziehen. Beide Aspekte sind eng miteinander verbunden, und eine allgemeine, in sich abgeschlossene Bewertung ist nur sehr schwer möglich. Aus diesem Grund ist es sinnvoll, den Systementwurf am Beispiel konkreter Anwendungsbereiche einzugrenzen. Vorab werden grundsätzliche Fragen bezüglich der Sensorik, der Algorithmen und der Hardwareplattform beantwortet. Insbesondere wird auf die Entscheidung eingegangen, Kameras auch im Nahbereich zu verwenden. Die Eigenschaften von Laser- und Radarsensoren werden in dieser Arbeit kurz aufgelistet und gegenüber gestellt. Im weiteren Verlauf werden diese jedoch nicht näher betrachtet.

Die von der Kamera erstellten Bilder können direkt der Hardware zugeführt werden, die die Stereoanalyse übernimmt. Voraussetzung dafür ist jedoch, dass die notwendige Vorverarbeitung entweder durch Logikressourcen des FPGAs oder durch zusätzliche Hardware wie zum Beispiel Signalprozessoren realisiert wird. Die ohnehin nur sehr begrenzt vorhandenen Ressourcen des FPGA würden erheblich durch die Vorverarbeitung eingeschränkt, so dass die Bearbeitung der Bilddaten durch eine GPU von Vorteil sein kann. Das gleiche gilt für die Nachbearbeitung, die je nach Komplexität von dem FPGA oder auch von der GPU ausgeführt werden kann. Ein Hostprozessor übernimmt üblicherweise die Ablaufsteuerung der gesamten Verarbeitungskette (Abb. 6.2).

Bildaufnahme

Neben den Kameras stehen messtechnische Geräte zur Verfügung, die durch Laufzeitmessung oder mittels Interferometrie Entfernung (Abb. 1.1) aktiv messen und hierbei

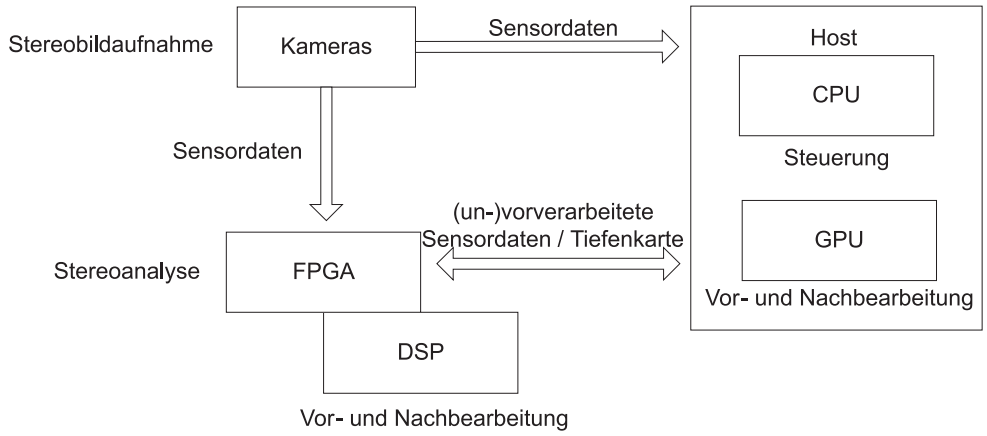


Abbildung 6.2.: Konzeptioneller Aufbau der Schnittstellen und Hardwarekomponenten.

punktweise eine bessere Genauigkeit als Kameras erreichen. Weiterhin haben Laserscanner und Radarsysteme eine Reihe weiterer Vorteile, die sie gegenüber passiven optischen Sensoren auszeichnen:

- Diese können in vollständiger Dunkelheit weiterhin Objekte detektieren.
- Der Kalibrieraufwand ist im Vergleich zu Stereoaufbauten geringer.
- Höhere Messgenauigkeit der Entfernung der Objekte bei zunehmenden Abstand.
- Aufgrund des Dopplereffekts können Radarsensoren zudem sehr genau die relative Geschwindigkeit der beobachteten Objekte messen.
- 3D-Laser-Scanner bieten gegenüber planaren Laser-Scannern den Vorteil, einen sehr weiten Bereich mit einer hohen Präzision zu erfassen. Exemplarisch stehen zwei 3D-Laser-Scanner wie zum Beispiel von Velodyne [35] und der Imager/Profiler [36], die die Umwelt durch Ablenkung eines Spiegels und Rotation auf einem Drehtisch einen Bereich von 360° horizontal und ca. 30° bzw. 310° vertikal erfassen. Beide Geräte unterscheiden sich in ihrer geometrischen Auflösung.

Demgegenüber stehen die systembedingten Nachteile der Laser- und Radarsensoren:

- Allgemein
  - Die gegenseitige Beeinflussung mehrerer aktiver Sensoren ist als problematisch anzusehen.
- Radar
  - Radarsensoren bieten eine relativ hohe Genauigkeit der Entfernung der Objekte, können jedoch diese nur sehr ungenügend geometrisch auflösen, so dass Informationen über Form und Größe des Objektes nicht abgeleitet

werden können.

- Laser
  - In der Praxis ist es häufig so, dass die Laserstrahlen stark gestreut und absorbiert werden, so dass nicht genug Energie zum Sensor zurückkehrt.
  - Neben den Reflexionen sind insbesondere die Messdaten von Laserscannern, die die Entfernung durch Überlagerung der Phasenverschiebung messen, stark verrauscht. Die Ursache dafür liegt in der Aufweitung des Laserstrahls, wodurch eine Punktmessung mehrere Objekte erfasst.
  - Aufgrund der geringen räumlichen Auflösung können 2D-Range-Scanner nicht sicher zwischen Gegenständen und zum Beispiel dem Anstieg der Fahrbahn unterscheiden. (Durch mehrere sich überlappende Scanner erhöht sich die Redundanz in der Hindernisserkennung, wodurch zum einen zwischen Weganstiegen und Hindernissen unterschieden werden kann, und zum anderen die Chance erhöht wird, ein Hindernis überhaupt zu erkennen.)

Kameras

Die heutigen optischen Sensoren, die vorrangig in der Industrie eingesetzt werden, sind in ihrer Bauform sehr kompakte Einheiten, die sich sehr gut in mobile Systeme integrieren lassen. Ihre relativ hohe Lichtempfindlichkeit und ihr hoher Dynamikumfang sind insbesondere für Innenraumapplikationen von Bedeutung. Neben der hohen räumlichen Auflösung im Megapixel-Bereich wird in Echtzeitanwendungen eine hohe zeitliche Auflösung gefordert. Dies steht prinzipiell im Widerspruch zur Sensitivität und dem räumlichen Auflösungsvermögen. Entweder werden weniger Pixel ausgelesen oder das SNR des Detektors verringert sich. Die Einführung von Mikrolinsen zur Verbesserung der Photoneneffizienz und die Empfindlichkeit im nahen Infrarotbereich erhöhen die Sensitivität des Sensors. Mikrolinsen über jedem Pixel bündeln das einfallende Licht auf die lichtempfindlichen Bereiche. Es entstehen jedoch Richtungsabhängigkeiten hinsichtlich des einfallenden Lichtstrahls. Je größer der Einfallswinkel ist, desto stärker nimmt die Lichtempfindlichkeit wieder ab.

Heutige Matrixkameras[1] erfassen zu einem Messzeitpunkt mehrere Megapixel an Grauwertinformation. Aufgrund der festen Matrixstruktur des Detektors existiert ein geometrischer Zusammenhang zwischen den einzelnen Pixeln (Messpunkten), der zusammen mit der inneren Orientierung dazu genutzt werden kann, die äußere Orientierung aus den Messpunkten algorithmisch zu bestimmen. Im Gegensatz dazu erfassen Laserscanner jeden Messpunkt einzeln und unabhängig voneinander, was sich insbesondere in dynamischen Umgebungen negativ auswirkt. Im Gegensatz zur Matrixkamera ist eine Neubestimmung der äußeren Orientierung des Laserscanners durch dessen Messergebnisse nicht möglich. Sie sind auf externe INS/GPS-Lösungen bzw. andere Trackingverfahren zur Bestimmung der äußeren Orientierung angewiesen, so

[1] Für Zeilenkameras gilt ähnliches.

dass sich Laserscanner für Navigationsaufgaben nur bedingt eignen. Des Weiteren besitzen heutige Industriekameras eine Ausgangsdatenrate von bis zu 50 Msamples/s. Der schnellste Laserscanner liefert maximal 4 Msamples/s. Weiterhin können Kameras Farbinformation liefern, die in einer Vielzahl hochwertiger Matching-Algorithmen benötigt werden.

Um eine Aussage über das optische Auflösungsvermögen eines Sensorkopfes treffen zu können, ist es hilfreich, die PSF und die MTF des optischen Systems zu bestimmen bzw. zu vergleichen. Die Punktverschmierungsfunktion stellt einen Zusammenhang zwischen Bild- und Objektpunkt dar und spiegelt die Impulsantwort des gesamten linear abbildenden Systems wieder. Zu dem Gesamtsystem gehören alle optischen Elemente wie zum Beispiel Filter, Linsen, Spiegel und Detektoren.

	Abtastrate	Reichweite	Genauigkeit/ Reichweite	Auflösung	Öffnungswinkel	Bauform
Radar	hoch	hoch	hoch	gering	klein (LR), groß (SR)	sehr kompakt
Laser	gering	hoch	hoch	mittel	klein (Lidar), groß (360°)	groß
Kamera	hoch	mittel	mittel	hoch	mittel-hoch	sehr variabel

Tabelle 6.1.: Gegenüberstellung von Radar, Laserscannern und Messkameras bzgl. der Abtastrate, Reichweite, Genauigkeit, Auflösung und Öffnungswinkel

Die Eigenschaften von Radar, Laserscannern und Kameras in Table 6.1 zusammengefasst.

Neben den Eigenschaften des Detektors hat die Optik ebenfalls einen entscheidenden Einfluss auf die Güte der Bildaufnahmen. Aufwendige und kostspielige Linsenkonstruktionen sind möglich, die die radiometrischen und geometrischen Verzeichnung weitestgehend kompensieren. Telezentrische Objektive weisen konstruktionsbedingt theoretisch keine geometrischen Verzeichnungfehler auf. Im Allgemeinen werden diese Objektive auf dem Gebiet der industriellen Automatisierung und im Speziellen bei der optischen Materialprüfung verwendet. Aus Kostengründen und mechanischen Gesichtspunkten wird jedoch auf telezentrische Objektive im Bereich der mobilen Robotik verzichtet, sodass die verwendeten kurz- und weitbrennweitigen Objektive geometrisch und radiometrisch kalibriert werden müssen.

Im Innenbereich sind weite Öffnungswinkel wünschenswert. Sie verzeichnen jedoch stark an den Rändern, auch ist eine verminderte Empfindlichkeit am Rand (Rand-

abfall) bei Weitwinkelobjektiven ebenfalls stärker gegeben. Für Applikationen, deren Sichtfeld sich auf bestimmte Szenen einschränkt, sind Objektive mit einem schmaleren Öffnungswinkel vorteilhafter. Typische Applikationen hierfür sind Aufnahmen entlang von Fahrbahnen oder Schienenwegen, bzw. entlang von festen Trajektorien.

Kalibrierung

Ein Standardverfahren zur Ermittlung der inneren und äußeren Parameter einer Matrixkamera basiert auf der Bildaufnahme einer bekannten Kalibriertafel. Mehrere Aufnahmen aus unterschiedlichen Betrachtungswinkeln müssen gemacht werden, um eine gute Konditionierung der Gleichungssysteme zu erhalten. Für langbrennweitige Kameraaufbauten ist die Kalibrierung mit Kalibriertafeln ungeeignet. Alternativ können sowohl Matrix- als auch Zeilenkameras in einem Goniometer/Kollimator-Aufbau geometrisch kalibriert werden, in dem für eine Teilmenge der Pixel die Blickrichtung bestimmt wird. Ein Stellmotor fährt die Pixelpositionen einzeln an. Aus der Kombination der Bewegungsdaten der Goniometermotoren und der belichteten Pixel können die inneren und äußeren Parameter der Kamera bestimmt werden [98].

Alternativ können die Kameraparameter mit Hilfe eines optischen diffraktiven Elements (DOE) bestimmt werden, wie in D. Griessbach et al. [40] vorgestellt. Aufnahmen mit einem DOE sind translations-invariant, wodurch eine getrennte Auswertung der Rotations- und Translationsbewegung möglich ist. Weitere Eigenschaften des Verfahrens sind:

- schneller Kalibriervorgang,
- hohe Messgenauigkeit in der Praxis, die mit Standardverfahren vergleichbar ist,
- relativ kleiner Aufbau,
- Möglichkeit der Nachkalibrierung im Feld.

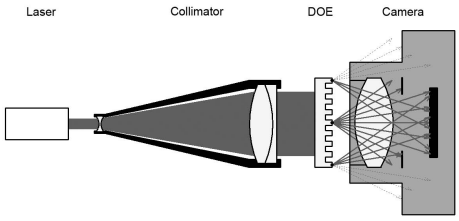
Abb. 6.3a veranschaulicht den Aufbau zur Bestimmung der Kameraparameter anhand eines DOEs. Eine Voraussetzung für ein genaues Messergebnis ist ein Laser, der einen Photonenstrahl konstanter Wellenlänge erzeugt. Ein Kollimator weitet den Strahl auf, so dass das gesamte DOE ausgeleuchtet wird. Die vom DOE erzeugte Gitterstruktur wird wiederum vom Detektor erfasst und als Einzelbild gespeichert. Die Parameter der inneren Orientierung K und der Verzeichnung $k_1 \dots k_n$ können mit einer Aufnahme berechnet werden.

Unter Berücksichtigung der Verzeichniskorrektur wird Equation 2.11 zu

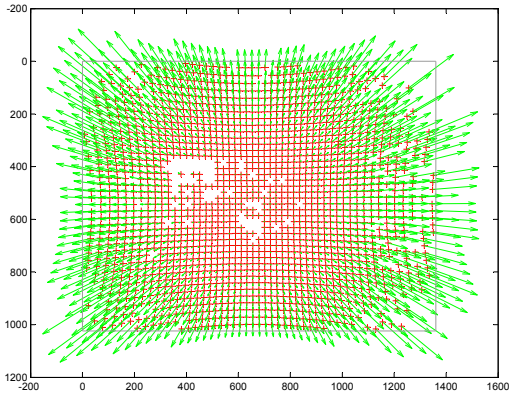
$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + f \begin{bmatrix} x \\ y \end{bmatrix} (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{6.1}$$

mit

$$r^2 = x^2 + y^2$$



(a) Kalibrierungssaufbau



(b) Verzeichniskorrektur

Abbildung 6.3.: Kamerakalibrierung mit Hilfe optischer diffraktiver Elemente (Quelle [40]).

erweitert, die wiederum nach den Parametern $m := (f, u_0, v_0, k_1, k_2, k_3, \omega, \varphi, \kappa, \alpha, \beta)$ für die Messpunkte $[\tilde{u}\ \tilde{v}]$ minimiert wird:

$$\min_m \left\| \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} - \begin{bmatrix} u \\ v \end{bmatrix} \right\|^2$$

mit:

- f, u_0, v_0 = innere Orientierung
- ω, φ, κ = äußere Orientierung der Kamera
- α, β = Verkippung des DOE zum Laser

Aufgrund dessen, dass Aufnahmen mit dem DOE translations-invariant sind, muss die Translation im Stereofall separat durch eine Kalibriertafel geschätzt werden. Die relative Rotation der beiden Kameras zueinander und die innere Orientierung wird bereits für beide Kameras durch die Messung mit dem DOE bestimmt.

Aus den Werten der inneren und äußeren Orientierung wird die Fundamentalmatrix berechnet, die Eingang in das Verfahren von A. Fusiello et al. [29] zur Rektifizierung der Stereobilder findet. Neben der radiometrischen und der geometrischen Kalibrierung muss die Synchronisation der Kamerapaare beachtetet werden. Beide Kameras müssen zur gleichen Zeit die Szene erfassen.

Vorverarbeitung

Temperatur und mechanische Beanspruchung können dazu führen, dass die ermittelten Werte aus der Kalibrierung ihre Gültigkeit verlieren. Insbesondere die Parameter

der relativen Orientierung müssen regelmäßig kontrolliert werden. H. Hirschmüller et al. [49] haben zu diesem Zweck Filteransätze untersucht, mit deren Hilfe die Einflüsse der Dekalibrierung minimiert werden können. Rangordnungsfilter wie der Sobel- und Census-Filter lassen sich relativ einfach in einem FPGA-Design integrieren. Es muss lediglich ausreichend Speicher entsprechend der Fenstergröße vorgehalten werden. Insbesondere bei der Verkettung von Filtern lassen sich diese effizient umsetzen.

Matchingverfahren

Eine Vielzahl unterschiedlicher Ansätze existiert, um zwischen mehreren Punkten Korrespondenzen zu finden (Abb. 2.11). Die qualitativ besten Disparitätenkarten werden durch eine globale Herangehensweise erzeugt. Diese sind für die echtzeitkritischen Anwendungen der Robotik jedoch ungeeignet, so dass in der Vergangenheit der Schwerpunkt auf den lokalen Ansätzen lag. Die Einbußen bei der Qualität der Ergebnisse mussten in Kauf genommen werden.

Erstmalig im Jahr 2006 veröffentlicht, stellt der am Deutschen Zentrum für Luft- und Raumfahrt entwickelte Matching-Algorithmus SGM eine sehr effiziente Alternative zu den klassischen Matchingverfahren dar. Aus nachstehenden Gründen wird der SGM-Algorithmus als Matching-Verfahren in das hier vorgestellte Konzept eingebunden.

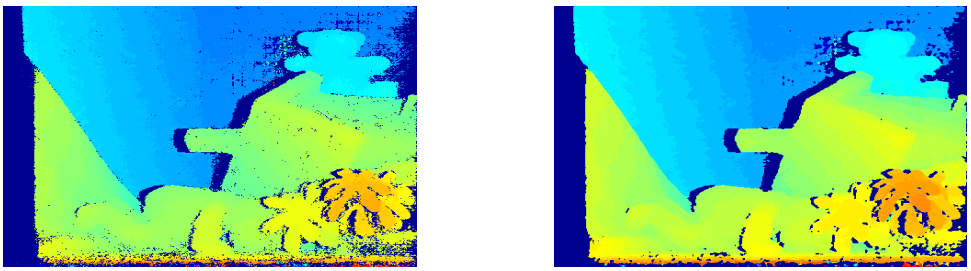
- Die Qualität der erzeugten Disparitätskarten ist denen der lokalen Verfahren überlegen [17] Stand: Juli 2010 (Abb. A.-4).
- Die Ausgabequalität von SGM [50] ist mit denen der globalen Ansätze vergleichbar.
- Die SGM-Variante Consistent-Semi-Global-Matching (cSGM) [48] gehört mit zu den besten Verfahren hinsichtlich Qualität [17] Stand: Juli 2010 (Anhang Abb. A.-4).
- Der SGM-Algorithmus [50] erlaubt es, abgeleitete Bildpaare zuzuordnen.
- Der SGM-Algorithmus wird für produktive Systeme in der Fernerkundung erfolgreich eingesetzt [51].
- Das Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt besitzt das Wissen und die Patente um den Algorithmus.

Aufgrund der aufgezählten sehr günstigen Eigenschaften des SGM-Algorithmus bietet dieser Algorithmus die notwendigen Voraussetzungen, um die allgemeinen Anforderungen aus section 6.1 und folgend zu erfüllen. Trotz seiner relativ geringen Komplexität im Vergleich zu den globalen Stereo-Korrespondenzverfahren ist der SGM-Algorithmus ein robuster Matcher für die Zuordnung von Bildern, die reale Bedingungen widerspiegeln. Insbesondere bei nicht-synthetischen oder nicht-simulierten Aufnahmen liefert der SGM-Matcher teilweise bessere Ergebnisse als globale Verfahren, die die Middlebury-Liste [17] anführen [112].

Dazu sind neben den in section 5.3 vorgestellten Arbeiten von H. Hirschmüller et al. [55] Untersuchungen zu den Auswirkungen radiometrischer Unterschiede im Bildpaar hinsichtlich der Kostenfunktion vorgenommen worden. Drei Stereo-Matcher (Summation der Kosten + WTA, SGM und eine GraphCuts-Implementierung) wurden zur Validierung in [55] verwendet. Insbesondere wurden Änderungen der Belichtungsdauer, der Ausleuchtung, Post-Processing-Schritte der Kamera, Randabfall der Optik und Rauschen simuliert. Das Ergebnis dieser Arbeiten untermauert die Leistungsfähigkeit des gewählten Verfahrens.

Nachbearbeitung

Zur Verbesserung des visuellen Eindrucks wird typischerweise ein nicht-linearer Filter zur Rauschunterdrückung auf das Ergebnis angewendet. Einzelne Ausreißerpixel werden relativ gut unterdrückt, ohne dabei die Kanten des Bildes zu sehr zu verschmieren. Adaptive Medianfilter haben gegenüber dem Standardmedian bessere Eigenschaften bei nur geringem Rauschanteil, dafür muss jedoch ein Mehraufwand in Kauf genommen werden, der durch die variable Fenstergröße des adaptiven Filters verursacht wird.



(a) vorher (b) nachher

Abbildung 6.4.: Medianfilter mit einem 3×3 Fenster auf ein bekanntes Stereobildpaar angewendet [17].

Für eine Umsetzung des Medianfilters in Hardware müssen ein Zeilenbuffer und freie Logik für den Sortiervorgang zur Verfügung stehen. FPGAs besitzen relativ wenig internen Speicher, jedoch reicht dieser, um auch größere Fensterfunktionen realisieren zu können. Das Hauptaugenmerk liegt demnach auf dem Sortieralgorithmus, der maßgeblich den Ressourcenverbrauch und die maximale Prozessrate bestimmt. Der intuitive Lösungsansatz für die Sortieraufgabe ist ein Komparator-Netzwerk, das dem Bubble-Sort-Algorithmus entspricht [101]. Dieser Ansatz skaliert jedoch sehr schlecht für große Fensterbereiche. Für ein $M \times N$-Fenster werden

$$\frac{M \cdot N - 1}{2} \cdot M \cdot N$$

Komparatoren benötigt. Für große Fenster ist ein Histogrammansatz besser geeignet

[23].

Diskussion

Die logischen Operatoren und deren Komplexität bestimmen die Verarbeitungsrate, mit der die Eingangsdaten innerhalb der Prozesselemente verarbeitet werden können. An dieser Stelle ist entscheidend, wie gut die Operationen auf die Hardware abgebildet werden können. Ziel ist es, einen hohen Prozesstakt zu erreichen, dies gilt besonders für Anwendungen, die die Echtzeitanforderung erfüllen müssen. Die Eingangsdaten müssen so schnell wie möglich verarbeitet werden, was durch möglichst viel parallel ausgeführte Operationen erreicht werden kann. Der Grad der Parallelität ist durch die zur Verfügung stehenden Hardware-Ressourcen begrenzt. Durch Duplizierung der Hardware kann möglicherweise die Anzahl der parallel ausgeführten Prozesse erhöht werden. Voraussetzung dafür ist, dass die Routinen separierbar sind und keine rekursiven Abhängigkeiten bestehen, die eine parallele Ausführung verhindern.

Zusammenfassend sind die nachfolgend aufgezählten Kriterien geeignet, die Leistungsfähigkeit des Systems zu charakterisieren:

- realisierbare Ausführungsgeschwindigkeit
  - Ein- und Ausgangsdatenrate
  - Prozessierungsrate
  - Zuverlässigkeit
- Ressourcenverbrauch
  - Energieverbrauch
  - physische Parameter wie Gewicht und Volumen
- Ausgabequalität
  - Messbereich
  - Genauigkeit im Messbereich

Die Gewichtung der eben genannten Merkmale muss aus den Anforderungen abgeleitet werden.

7. Ein neues echtzeitfähiges System für mobile Anwendungen

Nachdem im vorhergehenden Teil dieser Arbeit ein Konzept für ein echtzeitfähiges System zu Gewinnung von Tiefeninformation aus Stereobildern entwickelt worden ist, wird im nachfolgenden Abschnitt eine Realisierung für den Bereich der Robotik vorgestellt. Die Anforderungen an das System ergeben sich dabei spezifisch aus den in section 6.1 gezeigten Anwendungsfällen. Im Vordergrund der Realisierung steht dabei immer das Laufzeitverhalten bzw. die Echtzeitfähigkeit des Gesamtsystems. Das bedeutet, dass die Reaktionszeit des Systems zum einen den Anforderungen entsprechen muss und zum anderen das System in der gegebenen Zeitspanne ein verwertbares Resultat liefert. Dies wird für Robotikapplikationen in sicherheitsrelevanten Bereichen besonders deutlich. Hier wird die Echtzeitfähigkeit als kritisch angesehen. Ein optisches System zur Erkennung von Hindernissen, das potentielle Gefahren zu spät erkennt, ist nicht geeignet. Ähnliche Einschränkungen ergeben sich, wenn das System in der gegebenen Zeit reagiert, das Ergebnis jedoch so schlecht ist, dass daraus keine Rückschlüsse mehr gezogen werden können. In diesem Sinne wird eine strenge Echtzeitbedingung formuliert.

7.1. Anforderungen

Insbesondere für mobile Anwendungen wird deutlich, dass die Bildverarbeitung oftmals eine untergeordnete Teilaufgabe innerhalb eines komplexen Designs erfüllt, was wiederum relativ hohe Anforderungen bezüglich der Größe und der Energieaufnahme an das Subsystem stellt. Multiplikation der Hardware scheidet häufig als Option aus, so dass Echtzeitfähigkeit im verstärkten Maße durch die Algorithmik ermöglicht werden bzw. realisierbar bleiben muss.

Als Grundlage für weitere Betrachtungen werden folgende Anforderungen für das Gesamtsystem festgelegt:

- Das Gesamtsystem muss für mobile Anwendungen geeignet sein.
- Die Qualität der Tiefenkarte soll dicht und nach dem aktuellen Stand der Technik bestmöglich sein.
- Die optische Auflösung soll mindestens 640×480 Pixel betragen.
- Die Bildwiederholrate muss 25 Hz betragen.

- Die radiometrische Mindestauflösung beträgt 8 bit.
- Die geforderte Tiefenauflösung soll 128 Disparitätsstufen betragen.

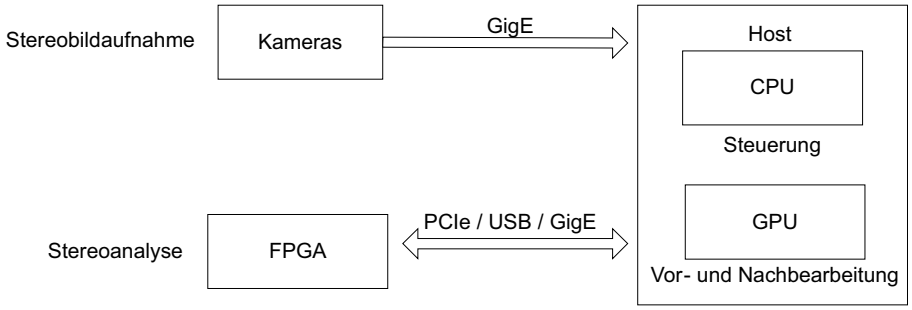


Abbildung 7.1.: Realisierung der Schnittstellen und der Hardwarekomponenten für Robotikanwendungen

In Abb. 7.1 ist der Basisaufbau für das Gesamtsystem skizziert. Je nach vorhandenen Ressourcen können Komponenten fehlen bzw. mehrfach vorhanden sein. PCIe, CameraLink und GigE-Vision sind etablierte Standardschnittstellen, die zur Übertragung der Kameradaten und der Tiefeninformation genutzt werden und die notwendige Ausgangsdatenrate der Kameras von ca. 15 MB/s

$$\text{Eingangsdatenrate} = \text{Bildgröße} \cdot \text{Bildwiederholrate} \cdot 2$$

erreichen.

7.2. Bildaufnahme

In Robotikanwendungen kommen relativ leistungsschwache CPUs zum Einsatz, die für die Vorverarbeitung (Entzeichnung und Rektifizierung) der Bilddaten in Echtzeit mit der oben genannten Auflösung nicht geeignet sind. Aus diesem Grund sollten diese Aufgaben wenn möglich von den oftmals vorhandenen Graphikprozessoren[1] übernommen werden. Eine Vorverarbeitung mit Hilfe des FPGAs ist ebenfalls möglich und grundsätzlich auf zwei Arten realisierbar. Zum einen können die Grauwerte des korrigierten Bildes online berechnet und zum anderen durch statische Look-Up-Tabellen [74] bestimmt werden. Der Vorteil der Online-Berechnung liegt im geringen Verbrauch von internen Speicherzellen. Diese Herangehensweise wird durch den hohen Anteil an notwendigen Logikressourcen wieder relativiert. Eine Lösung mit Hilfe von Tabellen hat den entgegengesetzten Effekt. Bei den im weiteren Verlauf verwendeten Objektiven muss mit einer Verzeichnung von bis zu 160 Pixeln gerechnet werden, so dass mindestens doppelt so viele Zeilen des Bildes um den aktuellen Pixel zwischengespeichert werden müssen (Abb. 7.2). Die Berechnung der korrigierten Stereobilder erfolgt indirekt. Im Falle eines modellbasierten Ansatzes wird in einem ersten Schritt die

[1] Die aktuelle Implementierung ist auf Nvidia-GPUs beschränkt.

Equation 6.1 gelöst. Durch Optimierungsschritte sind n-Rechenoperationen notwendig. Anschließend werden die Punkte

$$\begin{bmatrix} u \\ v \end{bmatrix}$$

mit der Transformationsmatrix H multipliziert. Zur Vermeidung von Treppen-Effekten bei der Entzeichnung und Rektifizierung werden die Ausgangsdaten bilinear interpoliert. Die Werte der Transformationsmatrix werden durch die geometrische Kalibrierung unter Ausnutzung des DOE-Verfahrens aus Table 6.2 ermittelt. Das Verfahren stellt eine Genauigkeit von mindestens $\frac{1}{2}$ Pixel gegenüber dem Modell sicher. Als Ka-

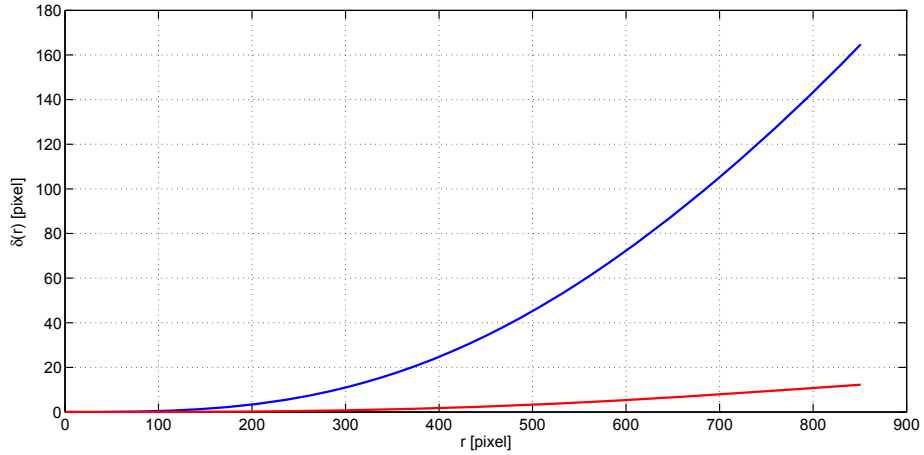


Abbildung 7.2.: Unterschiedlich starke Verzeichnungen baugleicher Objektive (SKN Cinegon-IR) für 4.8 mm und 12 mm Objektive. Legende: rot = 12 mm, blau = 4.8 mm Brennweite

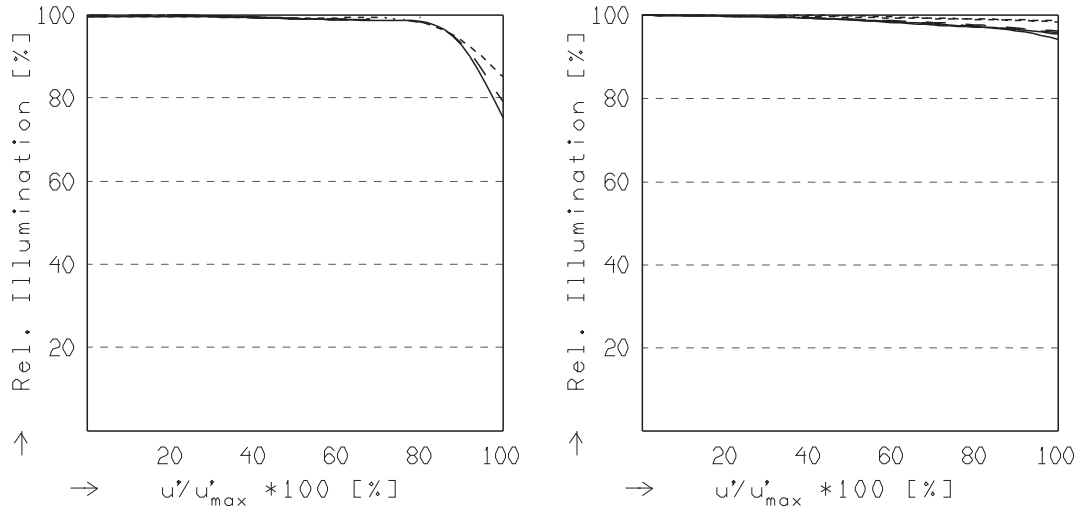
mera ist eine Standard-Industriekamera aufgrund des in ihr verbauten CCD-Detektors gewählt worden. Dieser weist eine in der Klasse der Industriekameras relativ hohe Lichtempfindlichkeit auf. Weitere Eigenschaften sind

- 1380 × 1024 Pixel
- 6.4 µm Pixelgröße
- Progressive-Scan
- Sony ICX-285 monochrom CCD
- 30 B/s bei voller Auflösung
- extern triggerbar
- relativ kompakte Bauform

Wahlweise werden Objektive mit einer Brennweite von 4.8 mm bzw. 12 mm verwendet, die entsprechend den Aufgaben vorrangig im Innen- bzw. im Außenbereich eingesetzt werden. Sowohl der Randabfall als auch die geometrische Verzeichnung für die Objektivserie Cinegon-IR von Schneider-Kreuznach sind in Abb. 7.2 gegenübergestellt.

Im Vergleich zu den 12 mm Objektiven, in Abb. 7.2 rot dargestellt, ist mit einer erheblich stärkeren Tonnenverzeichnung bei den Weitwinkelobjektiven, blau dargestellt, zu rechnen. Für Bildausschnitte im VGA-Format beträgt die radiale Verzeichnung des 4.8 mm Objektivs bereits ca. 25 Pixel.

Die verminderte Empfindlichkeit am Rand (Randabfall) ist bei Weitwinkelobjektiven ebenfalls stärker gegeben. Abb. 7.3a und Abb. 7.3b verdeutlichen diesen Unterschied. Sehr gut zu erkennen ist die schlagartige Dämpfung des einfallenden Lichtes zum Rand hin. Moderater fallen die Abschwächungen bei den 12 mm Objektiven aus (Abb. 7.3b).



(a) 4.8 mm Objektiv mit ca. 70° horizontalem Öffnungswinkel (SKN Cinegon-IR Datenblatt)

(b) 12 mm Objektiv mit ca. 40° horizontalem Öffnungswinkel (SKN Cinegon-IR Datenblatt)

Abbildung 7.3.: Unterschiedlich starker Randabfall baugleicher Objektive (SKN Cinegon-IR) für 4.8 mm und 12 mm Objektive. Legende: durchgezogene Linie = Fokus $\rightarrow \infty$; gestrichelte Linie = Fokus $\rightarrow 67.2\,\text{cm}$; gepunktete Linie = Fokus $\rightarrow 16.7\,\text{cm}$.

7.3. Stereo-Matching

Der SGM-Algorithmus lässt unterschiedliche Kostenfunktionen als ein Maß für die Ähnlichkeit zweier Pixel zu. In Abhängigkeit der Anwendung, können einzelne Module getauscht werden. Dieses Vorgehen wird durch die standardisierten Schnittstellen des Hardwarebetriebssystems unterstützt.

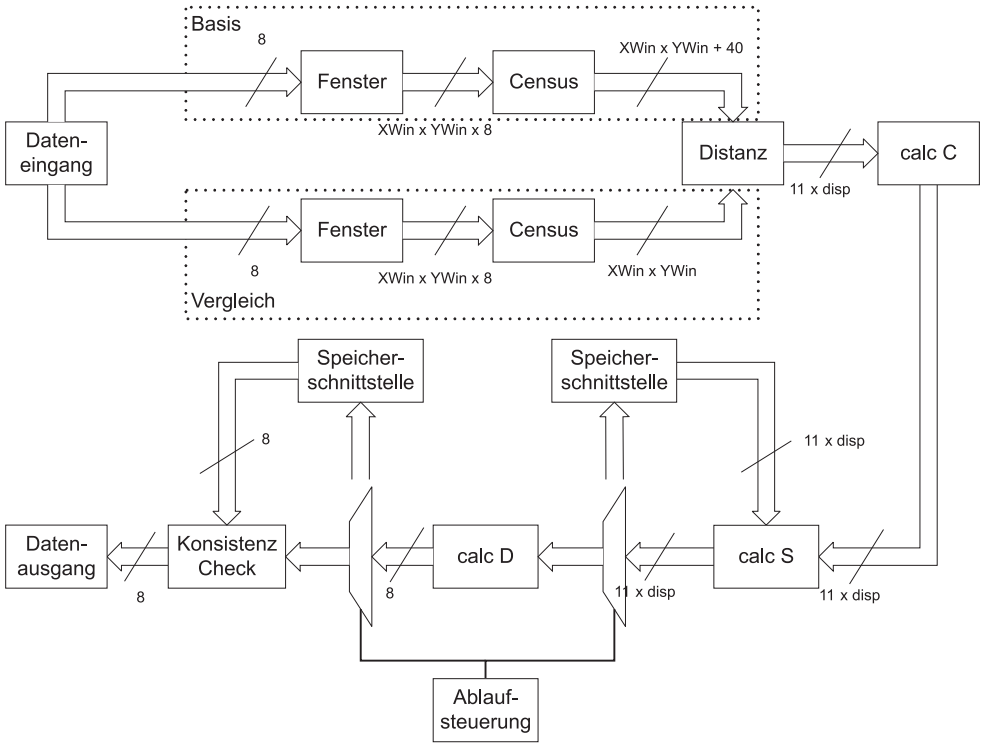


Abbildung 7.4.: Datenflussdiagramm des SGM-Algorithmus

Die Grauwerte sowohl des linken als auch des rechten Bildes werden in die doppelt vorhandene Verarbeitungskette zu Berechnung der Census-Bitvektoren eingegeben. In Abb. 7.4 werden die Pixel exemplarisch in 8 bit codiert. Aufnahmen mit größerem Auflösungsvermögen sind ohne weiteres möglich und haben keinen Einfluss auf die Wortbreiten zur Berechnung von S. Die Census-Transformation liefert Bitvektoren, deren Größe ausschließlich abhängig von der gewählten Fenstergröße ist. Das Kostenvolumen C wird in dem Modul `calc C` berechnet. Die Ähnlichkeit zweier Bitvektoren wird durch Bildung der Hamming-Distanz zwischen den Vektoren ermittelt. Die Hamming-Distanz kann relativ effizient in Hardware realisiert werden. Eine Fenstergröße von 5×5 Pixeln hat sich als optimal herausgestellt. In Abb. 9.5 ist der Einfluss der Fenstergröße für die Census-Transformation graphisch dargestellt. Als Fehlermaß wurde die mittlere quadratische Abweichung zur Groundtruth verwendet. Die Fenstergrößen variieren in der Größe von 3×3 bis 25×25 Pixel. Die Serie 2003 besteht aus den Bildern Tsukuba, Venus, Teddy und Cones. Sie spiegeln den Testsatz der Evaluierung von D. Scharstein et al. [97] wieder. Die Serie 2005 setzt sich aus den Aufnahmen Art, Books, Dolls, Laundry, Moebius und Reindeer zusammen. Die Groundtruth Daten stammen für beide Serien aus der Middlebury-Datenbank [17].

Die Bitvektoren werden mit der Exklusiv-Oder-Operation verknüpft. Ein Summationsoperator ermittelt die Anzahl der Bits, deren Wert Eins ist. Die Anzahl der not-

wendigen XOR-Vergleiche und Summationen ist abhängig von der Anzahl der Disparitätsstufen und der Fenstergröße. Neben der Hamming-Distanz werden die variablen Strafkosten P2 in dem Modul `calc C` berechnet und an das Modul zur Herleitung von S übergeben. Abb. 7.5 ist eine schematische Darstellung auf Logikebene der Equati-

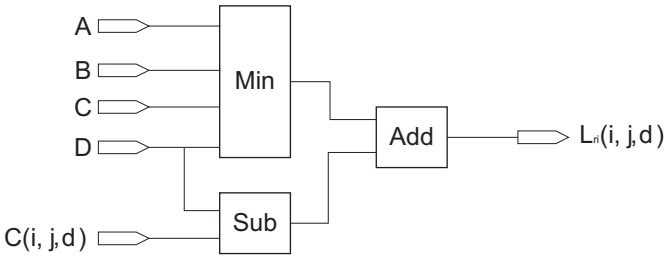


Abbildung 7.5.: Prozesselement für die Berechnung eines Eintrages in einer L-Säule

on 3.3 auf Seite 38. Equation 3.3 ist die Berechnungsvorschrift der Elemente einer so genannten L-Säule. Die Werte von A, B, C, D und $C(i, j, d)$ sind in 11 bit darstellbar und durch eine geringe Anzahl relativ einfacher logischer Operatoren miteinander verknüpft (Abb. 7.5). Als Ergebnis wird $L(i, j, d)$ übermittelt. Es sind lediglich drei Komparatoren, eine Addier- und eine Subtraktionseinheit notwendig, die alle mindestens 11 bit Daten verarbeiten können müssen. Der längste Signalpfad für einen Taktzyklus geht entlang zweier Komparatoren und zweier Addierer und ist damit relativ kurz.

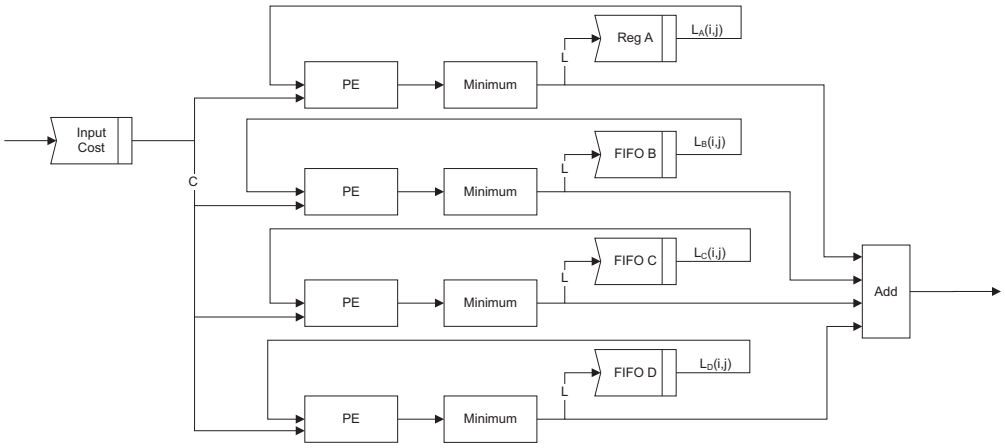


Abbildung 7.6.: Parallele Anordnung der Prozesselemente für die Berechnung eines Eintrages in S

Die Anordnung der Prozesselemente aus Abb. 7.5 in eine sequentielle (Abb. 7.7) und eine parallele Struktur (Abb. 7.6) hat direkten Einfluss auf den Ressourcenverbrauch und die Ausführungsgeschwindigkeit. Insbesondere die Anzahl der Komparatoren zur Realisierung der Minimumsuche über den gesamten Disparitätsbereich wirkt sich schnell auf den Verbrauch der zur Verfügung stehenden Logik aus. Der schematische Aufbau in Abb. 7.7 setzt jedoch eine komplexere Ablaufsteuerung voraus.

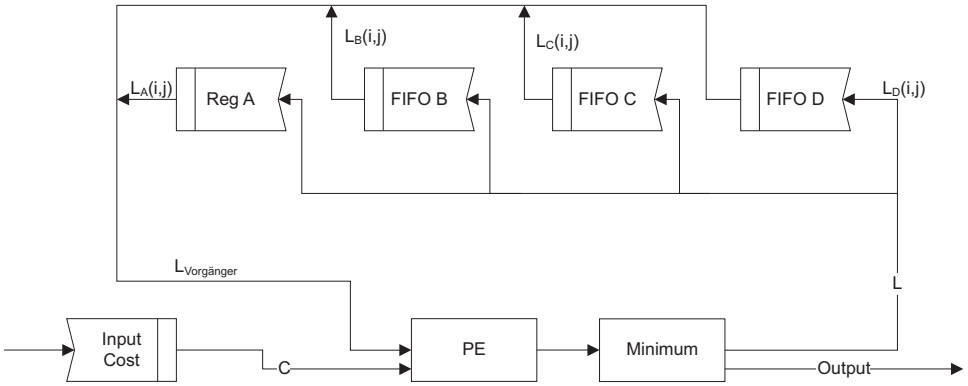


Abbildung 7.7.: Sequentielle Anordnung der Prozesselemente für die Berechnung eines Eintrages in S

Im Vergleich zu den kurzen Signalpfaden für die Berechnung aus Abb. 7.5 ist die Herleitung von

$$D = \min_k(L_r(i-r, j-r, k)) + P2$$

bezüglich der Minimumsuche komplexer. Die Aufstellung eines binären Suchbaumes führt zu einer Variante, die in der Ausführungsgeschwindigkeit optimal ist. Der Suchbaum besitzt $\log_2(Disparität)$ Stufen. Je mehr Stufen die Suche aufweist, desto langsamer wird der Prozesstakt, mit der die kombinatorische Schaltung betrieben werden kann. Register zwischen den Stufen können den Prozesstakt beschleunigen. Durch geeignetes Pipelining der Daten kann der gesamte Prozess beschleunigt werden.

Die Berechnung der Tiefenkarten aus dem aufsummierten Kostenvolumen S setzt ebenfalls eine Minimumsuche voraus, die aufgrund der nicht vorhandenen Abhängigkeiten ebenfalls durch Register in eine Pipeline-Struktur gebracht werden kann.

Einschränkungen des SGM-Algorithmus

Entsprechend den in section 7.1 definierten Anforderungen werden von der Systemleistung relativ hohe Bildwiederholraten erwartet. Bildgrößen von 640×480 Pixeln (VGA-Standard) und 128 Disparitätsstufen werden ebenfalls vorausgesetzt. Abb. 3.7 beschreibt schematisch den Datenfluss, der bei der Ausführung des SGM-Algorithmus möglich ist. Die Datenleitungen bzw. der Datenfluss zwischen den Dateneinund ausgängen, den Datenprozessierungspunkten und den notwendigen Zwischenspeichern sind mit römischen Ziffern I bis VI gekennzeichnet. Für die in Abb. 3.2 beschriebene Vorgehensweise, dass eine Tiefenkarte in VGA-Auflösung mit Hilfe des SGM-

Algorithmus in zwei Phasen berechnet wird, ergeben sich folgende Datenraten:

$$\begin{aligned}
\text{I} \leftarrow &\quad 640 \cdot 480 \cdot 2 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 14,6 \text{ MB}/s \\
\text{II} \leftarrow &\quad 640 \cdot 480 \cdot 2 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 14,6 \text{ MB}/s \\
\text{III} \leftarrow &\quad 640 \cdot 480 \cdot 2 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 128 \text{ disp} \cdot 2 \text{ Byte} \approx 3,7 \text{ GB}/s \\
\text{IV} \leftarrow &\quad 640 \cdot 480 \cdot 2 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 128 \text{ disp} \cdot 2 \text{ Byte} \approx 3,7 \text{ GB}/s \\
\text{V} \leftarrow &\quad 640 \cdot 480 \cdot 2 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 14,6 \text{ MB}/s \\
\text{VI} \leftarrow &\quad 640 \cdot 480 \cdot 2 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 14,6 \text{ MB}/s
\end{aligned}$$

Für Kacheln aus Luft- und Satellitenaufnahmen der Größe 2048×2048 und Disparitäten von bis zu 1024 Stufen kann das folgende Zahlenbeispiel die Anforderungen an die Schnittstellen und Berechnungsknoten veranschaulichen. Die Verarbeitung von Fernerkundungsdaten erfolgt im Normalfall offline, so dass eine Bildwiederholrate von 1 Bild/s angenommen wird, die im Bedarfsfall weiter gesenkt werden muss.

$$\begin{aligned}
\text{I} \leftarrow &\quad 2048 \cdot 2048 \cdot 2 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 8 \text{ MB}/s \\
\text{II} \leftarrow &\quad 2048 \cdot 2048 \cdot 2 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 8 \text{ MB}/s \\
\text{III} \leftarrow &\quad 2048 \cdot 2048 \cdot 2 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1024 \text{ disp} \cdot 2 \text{ Byte} = 8 \text{ GB}/s \\
\text{IV} \leftarrow &\quad 2048 \cdot 2048 \cdot 2 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1024 \text{ disp} \cdot 2 \text{ Byte} = 8 \text{ GB}/s \\
\text{V} \leftarrow &\quad 2048 \cdot 2048 \cdot 2 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 8 \text{ MB}/s \\
\text{VI} \leftarrow &\quad 2048 \cdot 2048 \cdot 2 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 8 \text{ MB}/s
\end{aligned}$$

Diskussion

Wie die vorhergehenden Rechenbeispiele erkennen lassen, sind die zu erwartenden Datenraten die schwerwiegenste Einschränkung. Das Ziel dieser Arbeit, ein echtzeitfähiges System zur Rekonstruktion der Tiefeninformation aus Stereobildpaaren zu realisieren ist mit den gegebenen Voraussetzungen bzgl. den Messgenauigkeiten, den Häufigkeiten der Messung, den vorhandenen Ressourcen und dem SGM-Algorithmus als Matching-Methode nicht realisierbar.

Eine weitere Einschränkung entsteht, wenn das Datenvolumen für S zu groß wird und nicht mehr vollständig in den externen bzw. Hauptspeicher abgelegt werden kann. Der letztere Fall tritt insbesondere bei der Auswertung im Bereich der Luft- und Satellitenfernerkundung auf. Die Anzahl der Elemente von S umfasst

$$\text{Höhe} \cdot \text{Breite} \cdot \text{Disparität}$$

Einträge. Die Speicherkapazität der integrierten SDRAM-Komponenten auf handelsüblichen FPGA-Karten ist ebenfalls sehr limitiert und bereits für Bilder in VGA-Auf-

lösung und mit 128 Disparitätsstufen nahe der Kapazitätsgrenze.

Wie in section 3.5 beschrieben, können die Eingangsdaten derart gekachelt werden, dass die benötigte Kapazität für die Speicherung des Volumens S ausreicht. Die Disparität wird für jede Kachel separat ermittelt und am Ende durch Überblenden mit den Nachbarkacheln zur Gesamtdisparität fusioniert. Ein Nachteil dieser Form der Kachelung besteht jedoch darin, dass es zu Artefakten an den Kachelgrenzen kommen kann, die sich im fusionierten Disparitätsbild zeigen. Je größer die Eingangsbilder sind, desto mehr Kacheln müssen erstellt werden, sodass die Artefaktbildung an dieser Stelle zunimmt. Jedoch wird bei der Bildung von Oberflächenmodellen, die zur Visualisierung verwendet werden, dieser Effekt praktisch nicht wahrgenommen [33].

Ein patentrechlich geschütztes Verfahren[2] von der Daimler AG umgeht die Artefaktbildung, indem zusätzlich zu den Volumen S der Kacheln die pfadweisen Kosten an den Kachelgrenzen für den gesamten Disparitätsbereich zwischengespeichert werden. Der Berechnungsaufwand verdoppelt sich, jedoch sind Artefakte ausgeschlossen. Mit diesem Verfahren wird ebenfalls der Bedarf an Arbeitsspeicher minimiert, jedoch zu Lasten der Rechenzeit. Der Gesamtbedarf an Arbeitsspeicher wird durch diesen Ansatz nicht minimiert und beträgt nach wie vor

$$8 \cdot \text{Disparität} \left(\frac{\text{Höhe} \cdot \text{Breite}}{n} + \frac{\text{Höhe} \cdot \text{Breite}}{m} \right) + (n+1) \cdot (m+1) \cdot \text{Disparität} \in O(n^3)$$

mit

- m, n = Kachelanzahl

Eine wesentliche Verbesserung wird nur durch eine Verringerung der gesamten Anzahl der Werte zur Zwischenspeicherung von S erreicht.

[2] DE 10 2008 017 834 A1

8. Der Efficient-Semi-Global-Matching-Algorithmus

Im vorhergehenden Kapitel ist deutlich geworden, dass die dort definierten Anwendungen und Anforderungen mit Hilfe der zur Verfügung stehenden Hardware und des SGM-Algorithmus nicht erfüllt werden können. Eine wesentliche Leistung dieser Arbeit besteht darin, einen Algorithmus zu entwerfen, mit dessen Hilfe es dennoch möglich ist, die gestellten Anforderungen zu erfüllen. Zu diesem Zweck wird im folgenden Kapitel ein neuer Algorithmus eingeführt, mit dessen Hilfe die Defizite des SGM-Algorithmus bezüglich Robotikanwendungen kompensiert werden können. Dieser Algorithmus wird im weiteren Verlauf efficient-Semi-Global-Matching-Algorithmus (eSGM) genannt und stellt eine speichereffiziente Form des Semi-Global-Matching-Algorithmus dar.

8.1. Motivation

Eine wichtige Voraussetzung für das sehr gute Abschneiden des SGM-Algorithmus besteht darin, dass keine voreiligen Entscheidungen bezüglich der Disparität getroffen bzw. einer bestimmten Richtung keine Präferenz gegeben wird. Für den SGM-Algorithmus ist es notwendig, dass alle Pfadkosten vollständig aufsummiert werden und im Anschluss das Kostenvolumen S nach der Disparität mit den minimalen Gesamtkosten durchsucht wird. In Abb. 8.1 sind die Teilkosten der acht unterschiedlichen Richtungen zu erkennen, die beispielhaft für eine Position im Bild entnommen worden sind. Der SGM-Algorithmus bestimmt den Wert 17 als Disparität aufgrund der an dieser Stelle geringsten Gesamtkosten.

In der Abb. 8.1 ist auch zu erkennen, dass sich das Gesamtkostenminimum genau an der Stelle des Minimums mindestens eines Kostenpfades befindet. Im Fall des Beispiels aus der Abb. 8.1 tritt dies bei sieben der acht Richtungen auf. In Bereichen, in denen keine Textur vorhanden ist, gestaltet sich die Suche nach korrespondierenden Punkten erwartungsgemäß sehr schwierig. Das von dem SGM-Algorithmus errechnete Gesamtkostenminimum deckt sich in dem Großteil der Fälle nicht mit der Disparität aus dem Referenzergebnis. Keine der acht Richtungen ist geeignet, eine eindeutige Vorhersage bzgl. der Disparität zu treffen, da die Konfidenz der Pfadkosten sehr gering ist. Beispiele, bei denen die vom SGM-Algorithmus gefundenen Disparitäten in solchen schwierigen Regionen mit der Referenz übereinstimmen, sind inhaltlich betrachtet zufälliger Natur. Der Zusammenhang entsteht dadurch, dass jeder Pfad Information bezüglich der kostenminimalen Disparität im Verlauf weiterpropagiert. Ein Minimum,

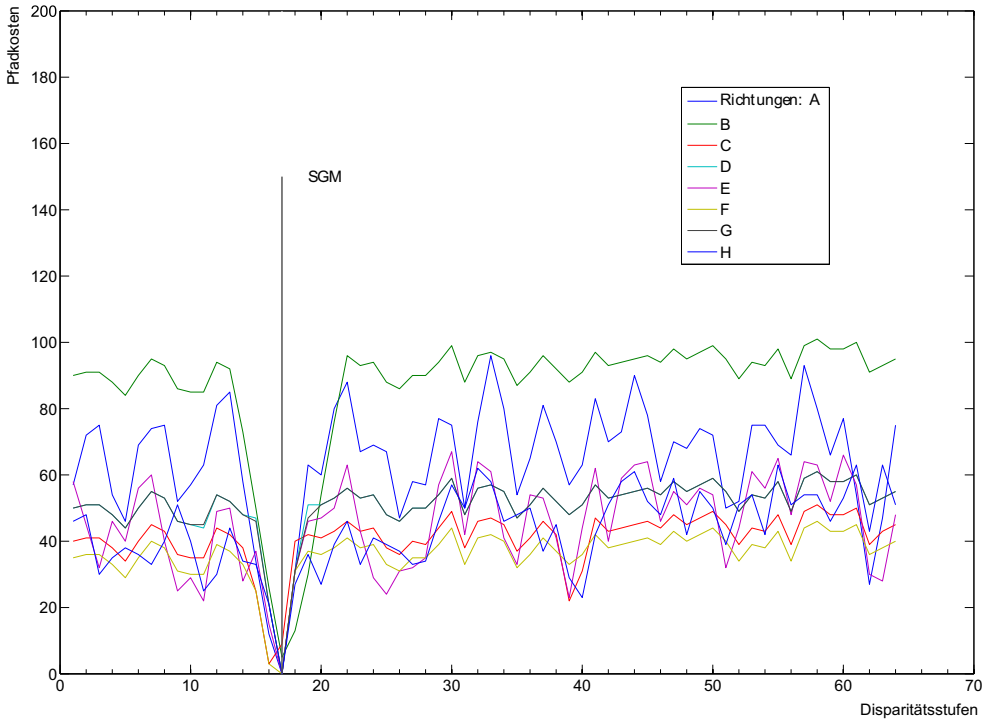


Abbildung 8.1.: Die einzelnen Pfadkosten für Abb. A.25 an der Position (100,75)

das zwar mathematisch korrekt gebildet worden ist, aber nicht durch mindestens einen der Richtungspfade abgebildet wird, muss inhaltlich als eine Fehlzuweisung bezüglich der Disparität interpretiert werden.

Der Suchbereich für die Disparität kann frühzeitig auf die Minima der Pfade reduziert werden, wodurch sich der Umfang des benötigten Zwischenspeichers stark reduziert. Zur Feststellung, welche Minimalstelle der einzelnen Pfade das gesuchte globale Minimum stützt, müssen nach wie vor die einzelnen Pfade akkumuliert werden, da ein globales Minimum eines Pfades nicht zwangsläufig dem theoretisch globalen Minimum aller Pfade entspricht.

Rein mathematisch ist diese Annahme nicht gerechtfertigt. Die Summe von Teilminima ist nicht zwangsläufig auch ein Gesamtminimum. Wie bereits erläutert wurde, ist es inhaltlich jedoch nicht sinnvoll, davon auszugehen, dass das Gesamtminimum sich an einer anderen Stelle befindet als das Minimum mindestens eines Pfades.

Damit ist ausgeschlossen, dass der eSGM-Algorithmus im Vergleich zu der Ausgangsvariante des SGM-Verfahrens den Suchbereich zu früh verkleinert. Die nachfolgende Fallunterscheidung fasst die oben genannten Unterschiede zwischen den beiden Algorithmen noch einmal zusammen.

- Wenn der SGM-Algorithmus einen Disparitätswert berechnet, an dessen Stelle sich auch ein lokales Minimum mindestens eines Pfades befindet, dann stimmen

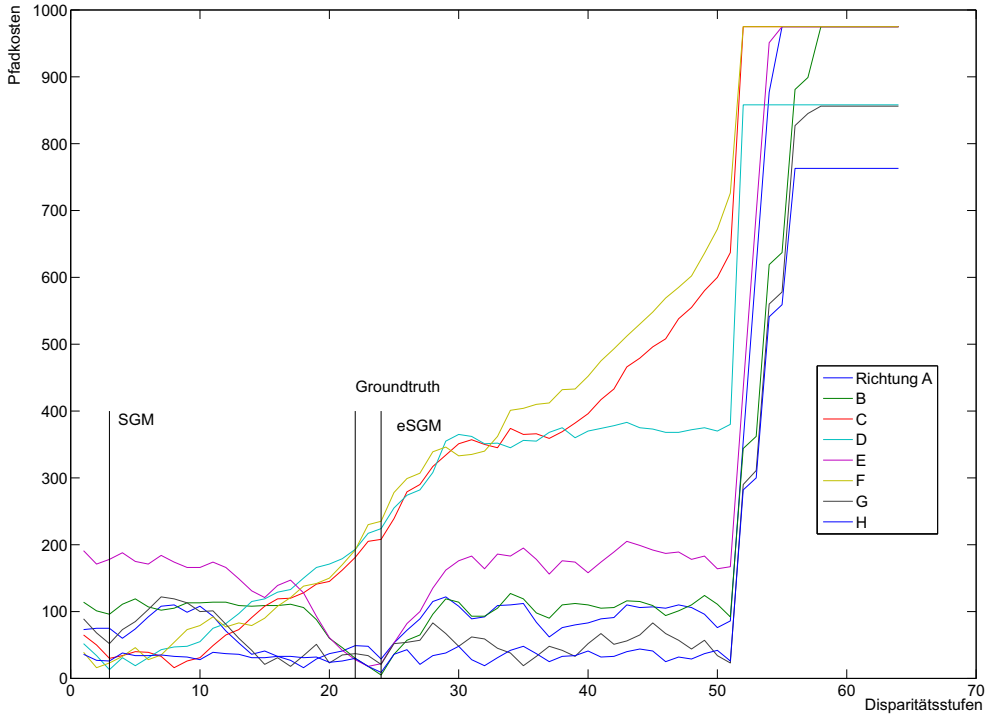


Abbildung 8.2.: Die einzelnen Pfadkosten für Abb. A.25 an der Position (400,75)

die Disparitäten von SGM und eSGM überein.

- Vorausgesetzt, der SGM-Algorithmus liefert einen Disparitätswert, an dessen Stelle sich kein lokales Minimum mindestens eines Pfades befindet, dann weicht entweder die gefundene Lösung von der Referenz ab oder ist zufällig identisch. Die Ergebnisse des eSGM- und SGM-Algorithmus müssen nicht übereinstimmen. Insbesondere müssen sie nicht mit der Referenz übereinstimmen.

Abb. 8.2 veranschaulicht den zweiten Punkt besonders deutlich. Sowohl der eSGM- als auch SGM-Algorithmus sind nicht in der Lage, eine zur Referenz identische Lösung zu finden. Die inhaltliche Abgrenzung zu der SGM-Variante wird ebenfalls deutlich. Die Richtungen B und E haben eine relativ starke Konfidenz bezüglich der Disparität, die sich in dem Ergebnis des eSGM-Algorithmus widerspiegelt und der Referenz sehr nahe kommt.

8.2. Der Algorithmus

Ausgehend von den obigen Überlegungen, ist eine Neuentwicklung des SGM-Algorithmus möglich, die bedeutend weniger Kapazität zur Speicherung von Zwischenergebnissen benötigt und von H. Hirschmüller, M. Buder und I. Ernst [53] im Jahre 2012 unter dem Namen eSGM vorgestellt worden ist. Die Veröffentlichung von M.

Buder [13] verweist auf [53] und entspricht inhaltlich dem Schwerpunkt dieser Arbeit. Statt die vollständigen Werte für alle Disparitätsstufen zu speichern, genügt es, lediglich die Stellen mit den jeweiligen Pfadkostenminima zwischenzuspeichern.

Der Basisalgorithmus benötigt zwei Pässe. Im ersten werden die vier Richtungen

$$A, B, C \text{ und } D$$

berechnet. Im zweiten Durchgang kommen noch einmal die Richtungen aus

$$E, F, G \text{ und } H$$

hinzu. Das vollständige Kostenvolumen S setzt sich somit aus zwei Teilen

$$S = S_1 + S_2$$

zusammen. Wobei

$$S_1(i,j,d) = \sum_{r \in \{A,B,C,D\}} L_r(i,j,d) \tag{8.1}$$

und

$$S_2(i,j,d) = \sum_{r \in \{E,F,G,H\}} L_r(i,j,d) \tag{8.2}$$

gilt. Das bedeutet, dass aus dem ersten Pass die jeweiligen Minima von vier Richtungen bekannt sind. $M_1 \dots M_4$ ist definiert als:

$$M_1(i,j) = S_1(I_1(i,j)) \tag{8.3}$$

$$\vdots$$

$$M_4(i,j) = S_1(I_4(i,j)) \tag{8.4}$$

mit

$$I_1(i,j) = \arg\min_d(L_A(i,j,d)) \tag{8.5}$$

$$\vdots$$

$$I_4(i,j) = \arg\min_d(L_D(i,j,d)). \tag{8.6}$$

In dem hier vorgestellten neuen Ansatz genügt es, sich auf die Kostenwerte $M_1 \dots M_4$ sowie deren Indizes $I_1 \dots I_4$ zu beschränken. Normalerweise müssen Höhe · Breite · Disparität-Kostenwerte zwischengespeichert werden. Durch den neuen Ansatz wird ein von der Anzahl der Disparitätsstufen unabhängiges Datenvolumen gespeichert.

Im ersten Durchgang des eSGM-Verfahrens, werden die vier Minima $M_1 \dots M_4$ und

die dazugehörigen Indizes $I_1 \dots I_4$ aus S_1 ermittelt. In einem zweiten Schritt können die vier gefunden Minima $M_1 \dots M_4$ mit den Einträgen aus

$$S_2(I_1) \dots S_2(I_4)$$

vervollständigt werden. Dadurch sind die Kostenwerte an den Stellen

$$I_1 \dots I_4$$

vollständig bekannt. Zusätzlich zu der Aktualisierung von $I_1 \dots I_4$ werden im zweiten Durchlauf auch die Indizes $I_5 \dots I_8$ für die übrigen Richtungen E, F, G, H ermittelt. Die Werte für $M_5 \dots M_8$ werden in analoger Art und Weise berechnet:

$$M_5(i,j) = S_2(I_5(i,j)) \tag{8.7}$$

$$\vdots$$

$$M_8(i,j) = S_2(I_8(i,j)) \tag{8.8}$$

mit

$$I_5(i,j) = \arg\min_d(L_E(i,j,d)) \tag{8.9}$$

$$\vdots$$

$$I_8(i,j) = \arg\min_d(L_H(i,j,d)) \tag{8.10}$$

Damit ebenfalls für diese Minimalstellen $I_5 \dots I_8$ die vollständigen Kosten bestimmt werden können, muss in einem dritten Durchgang S_1 erneut berechnet werden. Die Werte

$$S_1(I_5) \dots S_1(I_8)$$

werden mit

$$S_2(I_5) \dots S_2(I_8)$$

addiert und komplettieren die Kostenwerte.

Um den Speicheraufwand weiter zu minimieren, kann bereits nach dem zweiten Pass für $I_1 \dots I_4$ ein Teilminimum

$$\text{Min}_1 = \min(M_1 \dots M_4) \tag{8.11}$$

gebildet werden. Der zu Min_1 gehörige Indexwert muss ebenfalls gespeichert werden. Das zweite Teilminimum

$$\text{Min}_2 = \min(M_5 \dots M_8) \tag{8.12}$$

steht mit Berechnung des dritten Durchgangs zur Verfügung. Die Disparitätskarte

ergibt sich letztlich aus einer Minimierung der beiden Teilminima Min_1 und Min_2:

$$D = \min(\text{Min}_1, \text{Min}_2) \tag{8.13}$$

Komplexität des eSGM-Algorithmus

Hinsichtlich der Laufzeit, besteht zwischen dem SGM- und dem eSGM-Algorithmus kein Unterschied in der Komplexitätsklasse:

$$\mathcal{O}(W \cdot H \cdot d_{max} \cdot \text{Bildfrequenz}) \subseteq \mathcal{O}(n^4)$$

Genauso wie beim SGM-Algorithmus, müssen alle Pixel je nach Anzahl der Richtungen 4, 8 oder 16 mal besucht werden. Nach wie vor müssen entsprechend Equation 3.3 für jede Richtung und Pixel die Vektoren $L_A \ldots L_B$ von der Länge d_{max} betrachtet werden, so dass die Berechnung von Equation 3.3 in $\mathcal{O}(4 \cdot L) \subseteq \mathcal{O}(n)$ liegt. Für jede der vier Richtungen A, B, C, D und deren Komplement im Rückwärtsschritt A^*, B^*, C^*, D^* muss Speicher in der Größenordnung von

$$\mathcal{O}((W \cdot d_{\max} \cdot 3 + d_{\max}) \cdot 2 \text{ Bytes}) \subseteq \mathcal{O}(n^2 + n)$$

zur Verfügung gestellt werden. Anders als im Kernalgorithmus, wird das Pfadkostenvolumen S nicht mehr vollständig gespeichert. Der gesamte Speicherbedarf des eSGM-Verfahrens reduziert sich von $\mathcal{O}(n^3)$ auf

$$\mathcal{O}(W \cdot H \cdot 8) \subseteq \mathcal{O}(n^2)$$

Die Anforderungen an die Datenraten können demzufolge ebenfalls erheblich gesenkt werden. Für die in Abb. 3.2 beschriebene Vorgehensweise, dass ein VGA-Bildpaar phasenweise durchlaufen wird, ergeben sich folgende theoretischen Datenraten:

$$\begin{aligned}
\text{I} \leftarrow &\quad 640 \cdot 480 \cdot 3 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 22 \text{ MB}/s \\
\text{II} \leftarrow &\quad 640 \cdot 480 \cdot 3 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 22 \text{ MB}/s \\
\text{III} \leftarrow &\quad 640 \cdot 480 \cdot 3 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 2 \cdot 4 \cdot 1 \text{ Byte} \approx 176 \text{ MB}/s \\
\text{IV} \leftarrow &\quad 640 \cdot 480 \cdot 3 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 2 \cdot 4 \cdot 1 \text{ Byte} \approx 176 \text{ MB}/s \\
\text{V} \leftarrow &\quad 640 \cdot 480 \cdot 3 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 22 \text{ MB}/s \\
\text{VI} \leftarrow &\quad 640 \cdot 480 \cdot 3 \text{ Phasen} \cdot 25 \text{ B/s} \cdot 1 \text{ Byte} \approx 22 \text{ MB}/s
\end{aligned}$$

Gegenüber dem ursprünglichen SGM-Algorithmus erhöhen sich die Datenraten an den Schnittstellen I,II,V und VI um ein Drittel aufgrund des notwendigen dritten Passes, der neu hinzugekommen ist. Es ist jedoch eine erhebliche Reduzierung der erforderlichen Bandbreiten an den Schnittstellen III und IV zu beobachten. Die Speicheranbin-

dung ist nicht mehr abhängig von der Disparität und erschließt damit Potential für größere Disparitätsbereiche. Für Kacheln aus Luft- und Satellitenaufnahmen der Größe $2k \times 2k$ und Disparitäten von bis zu 1024 Stufen kann das ursprüngliche Zahlenbeispiel ebenfalls aktualisiert werden. Aufgrund der Offlineprozessierung von Fernerkundungsdaten kann die Bildwiederholrate von 1 Bild/s im Bedarfsfall weiter gesenkt werden.

$$\begin{aligned}
\text{I} \leftarrow &\quad 2048 \cdot 2048 \cdot 3 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 12 \text{ MB}/s \\
\text{II} \leftarrow &\quad 2048 \cdot 2048 \cdot 3 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 12 \text{ MB}/s \\
\text{III} \leftarrow &\quad 2048 \cdot 2048 \cdot 3 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 2 \cdot 4 \cdot 1 \text{ Byte} = 96 \text{ MB}/s \\
\text{IV} \leftarrow &\quad 2048 \cdot 2048 \cdot 3 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 2 \cdot 4 \cdot 1 \text{ Byte} = 96 \text{ MB}/s \\
\text{V} \leftarrow &\quad 2048 \cdot 2048 \cdot 3 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 12 \text{ MB}/s \\
\text{VI} \leftarrow &\quad 2048 \cdot 2048 \cdot 3 \text{ Phasen} \cdot 1 \text{ B/s} \cdot 1 \text{ Byte} = 12 \text{ MB}/s
\end{aligned}$$

Im Vergleich zum Basisalgorithmus kann durch eine Erhöhung des Berechnungsaufwandes um ein Drittel die notwendige Datenrate der Schnittstellen III und IV im Verhältnis 1:80 gesenkt werden. Besonderes Augenmerk gilt jedoch dem Speicherbedarf für zum Beispiel UltraCam-Aufnahmen. Durch Kachelung ist es mit dem SGM-Algorithmus möglich, UltraCam-Bilder in der Größenordnung von $11500 \cdot 7500$ Pixel zu berechnen. Rechenmaschinen mit 160 GB frei adressierbarem Speicher sind nur sehr schwer zugänglich. Der Algorithmus eSGM senkt den Speicherbedarf für die gleiche Ausgangskonfiguration auf circa 1,3 GB. Ultra-Cam-Daten sind damit ohne weiteres durch zum Beispiel Standard-PCs verarbeitbar.

Der aktualisierte Speicherbedarf ist in der Tabelle Table 8.1 für VGA-Auflösung, das UltraCam-System und einer beispielhaften $1024 \cdot 1024$-Industriematrixkamera (Gen) zusammengefasst.

Kamera-system	Auflösung	Speicherbedarf für L	Speicherbedarf für S (eSGM)	Speicherbedarf für S (SGM)
VGA	$640 \cdot 480$	241 KB	$\approx$ 2,3 MB	$\approx$ 33 MB
Gen	$1024 \cdot 1024$	384 KB	$\approx$ 8 MB	$\approx$ 1 GB
UltraCam	$11500 \cdot 7500$	66 MB	$\approx$ 1,3 GB	$\approx$ 160 GB

Tabelle 8.1.: Speicherbedarf für eSGM und SGM im Vergleich.

Subpixelverfeinerung

Durch die nicht mehr benötigte Speicherung des vollständigen Kostenvolumens S fällt die Möglichkeit weg, die Subpixelinterpolation nach Equation 3.8 direkt aus S abzuleiten. Die benachbarten Werte S_{-1} und S_{+1} von $d(i, j)$ stehen somit nicht zur Verfügung. Eine Möglichkeit zur Lösung dieses Problems besteht darin, den eSGM-Algorithmus

zu erweitern. Zusätzlich zu den Kostenwerten $S_1(I_r),\ r \in (1 \dots 8)$ werden die Werte der direkten Nachbarn $S_1(I_r^{\pm 1})$ mit

- $S_1(I_r)$ = Kostenwert aus S an der Stelle I_r aus Richtung r für den ersten Pass
- I_r = gewählter Index der Richtung r
- $I_r^{\pm 1}(i,j) = I_r(i,j) \pm 1$

zwischengespeichert. Der zusätzliche Speicherbedarf beschränkt sich lediglich auf acht weitere Einträge, so dass die Komplexität von eSGM unverändert gut bleibt.

8.3. Erweiterungen

Aufgrund der relaxierten Anforderungen an die Schnittstelle des externen Speichers entwickelt sich nun die Berechnung von Equation 3.3 zum begrenzenden Faktor für das Gesamtsystem. Insbesondere die Pipelinetiefe zur Bestimmung des Minimums bzw. Index über eine vollständige Säule L hat Einfluss auf die Rate, mit der die Bilder verarbeitet werden können. Die Herleitung der Kosten aus den Richtungen B, C und D ist unkritisch, da die Ergebnisse erst mit Beginn der nächsten Zeile benötigt werden. Lediglich das Minimum aus der Vorgängerrichtung A muss zum nächsten Pixeltakt zur Verfügung stehen. Aufgrund der maximalen Pipelinetiefe von $\log_2(disp)$ Stufen sind für 64 Disparitäten bereits sechs Taktzyklen notwendig. Um 25 VGA-Bilder pro Sekunde verarbeiten zu können, ist für den eSGM-Algorithmus ein Pixeltakt von ca. 40 ns erforderlich.

Unter der Annahme, dass der realisierbare Systemtakt 100 MHz beträgt, verlangt die Realisierung von `calc S` in Form eines sequentiellen Aufbaus wie in Abb. 7.7, dass die Berechnung der Kosten einer Richtung in einem Takt erfolgt. Daraus ergibt sich, dass die Bestimmung des Minimums aus der Richtung A in vier Takten abgeschlossen sein muss. Dieses Ziel ist unter den gegebenen Voraussetzungen für 64 Disparitätsstufen nicht zu erreichen. Zur Lösung dieser Problematik kann zum einen die Anzahl der Registerstufen reduziert oder zum anderen der Systemtakt erhöht werden. Beide Veränderungen haben einander entgegengesetzte Auswirkungen und schließen einander aus. Eine Verringerung der Registerstufen erzwingt eine Verlangsamung des Systemtakts im gleichen Maße wie eine Erhöhung des Systemtakts zusätzliche Registerstufen erforderlich macht.

Eine Adaption des Basisalgorithmus ist notwendig, die als eSGM* bezeichnet wird, um die geforderte Bildwiederholrate zu ermöglichen. In durchgeführten Untersuchungen hat sich gezeigt, dass nur eine minimale Verschlechterung zu beobachten ist, wenn anstelle des Minimums für die Richtung A das Minimum aus einer der drei verbliebenen Richtung herangezogen wird. Im nachfolgenden chapter 9 wird beim Vergleich der verschiedenen Algorithmen auch der eSGM* einbezogen.

9. Ergebnisse

Im folgenden Abschnitt wird die in chapter 7 vorgestellte Realisierung des Systems für die Rekonstruktion der Tiefeninformation aus Stereobildpaaren für verschiedene Hardwareplattformen anhand der Anforderungen, die in section 7.1 genannt wurden, diskutiert. In Anlehnung an die Übersicht aus Abb. 6.2 sollen folgende Aspekte hervorgehoben werden: Ressourcenverbrauch, Ausführungsgeschwindigkeit und die erreichbare Qualität der Disparitätskarte.

9.1. Laufzeitverhalten des Systems

Ein wesentlicher Aspekt dieser Arbeit basiert auf dem Einsatz von rekonfigurierbarer Hardware. Verwendung finden zwei FPGAs vom Typ Xilinx XC5VSX95T und XC5VLX110T, die im mittleren Leistungssegment angesiedelt sind. Neben diesen werden FPGAs vom Typ XC6SLX150 verwendet, die für energie- und preiskritische Anwendungen vorgesehen sind. Vorrangig unterscheiden sich beide Typenklassen durch Preis und Leistungfähigkeit.

Laufzeitverhalten des SGM-Algorithmus

Im Rahmen des standort-übergreifenden Forschungsvorhabens Echtzeit3D bestand die Möglichkeit, neben FPGAs weitere Plattformen zu validieren. Unter anderem wurden CPUs, Cell B/E und GPUs für das Stereo-Matching mit Hilfe des SGM-Algorithmus untersucht. Die Ergebnisse zu den CPU-Implementierungen basieren auf Arbeiten von Heiko Hirschmüller.[1] Cell B/E Systeme wurden von Gordon Cichon[2] analysiert und GPU-Systeme von I. Ernst et al. [22].

Die Ausführungsgeschwindigkeit ist stark abhängig von der Implementierung und den Fähigkeiten des Programmierers. Nichtsdestotrotz können einige qualitative Aussagen hinsichtlich der Ausführungsgeschwindigkeit in Abhängigkeit von der Hardwareplattform getroffen werden. Als Referenz dient die CPU-Implementierung *CPU1*. Sie basiert auf einer Mehrkern-Prozessor-Architektur von Intel (Xeon 2.66 GHz) und wurde in C realisiert. Die Erweiterung der CPU1-Implementierung durch die Verwendung des SSE2-Befehlssatzes führt zu einer optimierten CPU2-Version. Eine NVIDIA GPU Ultra 8800 wurde mit der Erweiterung OpenGL/Cg ebenfalls für eine SGM-Portierung verwendet. Die von Gordon Cichon untersuchte Hardware basiert auf Cell-Prozessoren,

[1] Quellen: Zitiert aus nicht veröffentlichter DLR-interner Kommunikation. heiko.hirschmueller@dlr.de
[2] gordoncichon@gmail.com

die in der Spielekonsole PS3 von Sony verbaut wurden; dabei konnten lediglich vier der acht vorhandenen Verarbeitungseinheiten (SPUs) verwendet werden, so dass hier auf eine suboptimale Lösung Bezug genommen wird.

CPU1 Xeon 2.66 GHz, Census, adaptive P2-Strafkosten entsprechend Equation 3.4, vollständiger LR-Check zur Konsistenzüberprüfung

CPU2 Xeon 2.66 GHz, SSE2, Census, adaptive P2-Strafkosten entsprechend Equation 3.4, vollständiger LR-Check zur Konsistenzüberprüfung

GPU1 8800 Ultra, OpenGL/Cg unter Linux

FPGA1 Xilinx Virtex 5, XC5VSX95T, 64 MB DDR2 SDRAM (component), 256 MB DDR2 SDRAM (SODIMM socket), Census, 125 MHz Systemtakt

Cell1 Sony Playstation 3, 4 SPUs in Nutzung, 256 MB RAM, Yellow Dog Linux 6.0, IBM Cell SK 3.0 in C, SAD als Matchingkosten, adaptive P2-Strafkosten entsprechend Equation 3.4,

Im folgenden Teil wird für die Laufzeitmessung ein Disparitätsbereich von 128 Stufen und eine VGA-Auflösung der Eingangsbilder angenommen (Abb. 9.1). Mit Ausnahme der Cell-Plattform wird Census als Kostenmaß verwendet. Von der Komplexität bezüglich der Ausführungsgeschwindigkeit her betrachtet, sind beide Kostenmaße gut vergleichbar, so dass dieser Umstand der Untersuchung nicht schadet.

Das beste Ergebnis hinsichtlich der Laufzeit liefern sowohl die GPU- als auch die FPGA-Lösung. Obwohl ein FPGA im Vergleich zu einer GPU ein größeres Potential hinsichtlich der Parallelisierbarkeit besitzt, kann der FPGA zunächst keinen Vorteil daraus ziehen. Die Ursache dafür liegt in dem sehr hohen Bedarf an Speicherbandbreite des SGM-Algorithmus, der von den getesteten Systemen nicht zur Verfügung gestellt werden konnte. Den überwiegenden Teil der Zeit verbleiben die Logikressourcen deshalb ungenutzt. Die Laufzeitangabe des FPGA1 basiert auf simulierten Werten, da nur eine Messgenauigkeit von 512×512 Pixeln mit 64 Disparitätsstufen mit der zur Verfügung stehenden FPGA-Konfiguration realisierbar ist. Die Skalierbarkeit des Systems rechtfertigt jedoch die oben getroffenen Aussagen.

Laufzeitverhalten des eSGM-Algorithmus

Erwartungsgemäß verdoppelt sich die notwendige Rechenzeit für die CPU-Implementierung. Der Vorteil des eSGM-Algorithmus ist für CPU-basierte Systeme somit in der Möglichkeit zu sehen, große Bildverbände ohne Kachelung zu berechnen. In Abb. A.-51 ist eine Satellitenaufnahme des Stadtzentrums von Berlin der Größe 9782×14580 Pixel zu sehen. Der notwendige Disparitätsbereich beträgt 512 Pixel (Abb. A.-52). Mit Hilfe des eSGM-Algorithmus kann das Bild vollständig in einem Schritt verarbeitet werden. Die Zwischenspeicherung benötigt lediglich 4.8 GB externen Speicher. Die Berechnung dieses Bildes durch den SGM-Algorithmus benötigt 272 GB und kann nur durch die in section 3.5 vorgestellte Kachelung realisiert werden.

Sowohl der eSGM- als auch der SGM-Algorithmus zuzüglich der Kachelung liefern qualitativ gleichwertige Ergebnisse. Es muss jedoch beachtet werden, dass in Bildsequenzen mit wenig Textur bzw. Aufnahmen mit einer besseren Bodenauflösung der Mehraufwand hinsichtlich der Kachelung steigt. Gleichzeitig muss die Maximalzahl der Disparitätsstufen für Regionen mit starken Höhenunterschieden wie zum Beispiel Gebirgen erweitert werden. Mit Hilfe des eSGM-Verfahrens können diese erweiterten Anforderungen grundsätzlich erfüllt werden.

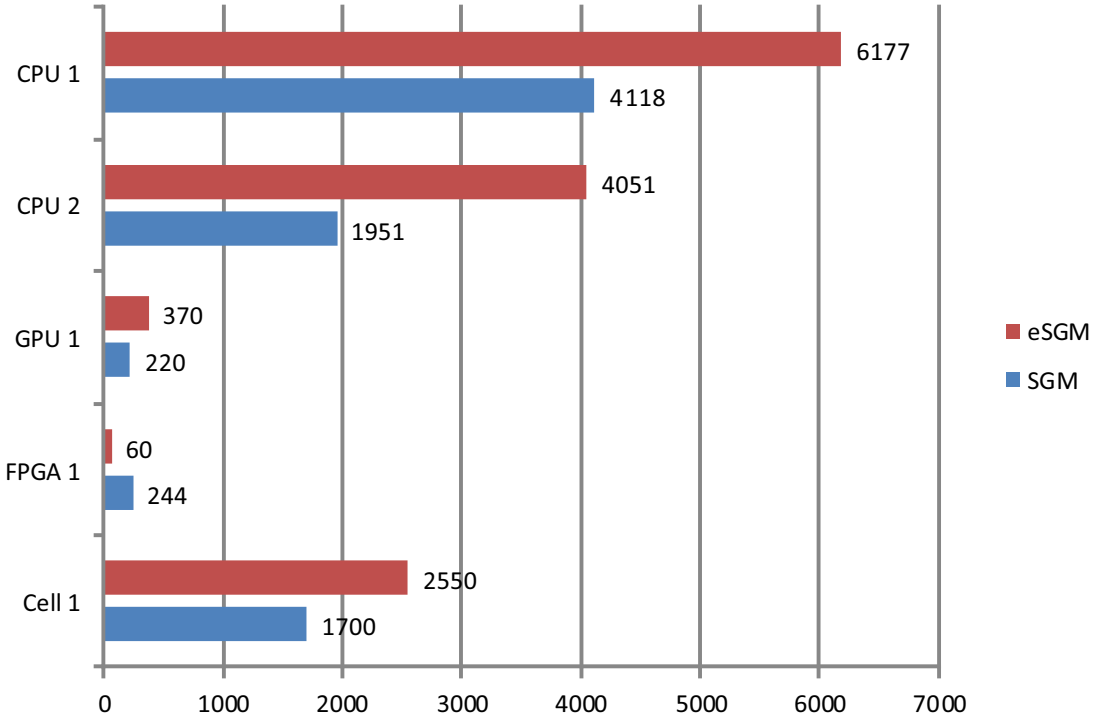


Abbildung 9.1.: Laufzeiten in ms. (Für CPU1 und Cell1 wurden die Werte bzgl. eSGM abgeschätzt.)

Die reduzierten Anforderung an die Speicherkapazität führen dazu, dass GPU-Implementierungen in die Lage versetzt werden, größere Bilddimensionen von bis zu 2024×2024 Pixeln und 1000 Disparitätsstufen verarbeitet zu können, die mit der SGM-Variante nicht möglich sind. Die Berechnung der eben genannten Konfiguration mit Hilfe einer GPU kann damit ca. 5,6-mal so schnell durchgeführt werden, als es mit einer optimierten CPU-Lösung möglich wäre [53]. Die Einsatzmöglichkeit der GPU-eSGM-Variante liegt somit im Bereich der Geofernerkundung mit entsprechender Kachelgröße.

Das Verhältnis zwischen der potentiell zur Verfügung stehenden Rechenleistung aktueller FPGA-Hardware und der Speicherbandbreite externer Speichermodule ist verantwortlich für die Leistungssteigerung, die mit dem eSGM-Algorithmus erzielt werden kann. Durch eine Verlagerung des Aufwandes von den vielen Datentransferoperationen

des SGM-Algorithmus hin zu mehr Rechenoperationen seitens des eSGM-Algorithmus wird das Gesamtsystem optimiert. Der Mehraufwand des eSGM-Algorithmus von 50 % zusätzlichen Berechnungen erweist sich somit im Vergleich zum SGM-Algorithmus nicht als Nachteil.

Insbesondere für FPGA-Lösungen ist eine erhebliche Verbesserung des Laufzeitverhaltens zu beobachten. Die Berechnung eines 640×480 Pixel großen Tiefenbildes benötigt ca. 30 ms. Die Überprüfung der Konsistenz erfordert eine vollständige Neuerstellung der Tiefenkarte, so dass sich die Laufzeit halbiert und 16,5 B/s realisiert werden können.

9.2. Skalierbarkeit

Im nachfolgenden Teil sollen die Skalierbarkeit des Systems bezüglich der drei Faktoren Ressourcen, Messgenauigkeit und Laufzeitverhalten untersucht werden.

Einfluss der Ressourcen

Eine Vergrößerung der Anzahl an Logikressourcen beeinflusst die Ausführungsgeschwindigkeit des Stereo-Matching-Algorithmus. Neben einer Verkürzung der Latenz des Systems könnte im besten Fall für jede Pixelposition nach $\text{Taktanzahl}_{\text{Minimum}} + \text{Taktanzahl}_{\text{PE}}$ Takten ein Wert bestimmt werden. Dies setzt voraus, dass die notwendigen Module vierfach ausgelegt werden und wie in Table 9.2 parallel berechnet werden. Table 9.1 stellt den zeitlichen Ablauf der Pfadkostenberechnung in einer Pipelinestruktur dar, wodurch vier zusätzliche Takte je Pixelposition notwendig sind. Unter der Annahme, dass die Pipelinestruktur zu jedem Zeitpunkt mit neuen Werten gefüllt werden kann, ermöglicht ein Systemtakt von 100 MHz einen Pixeltakt von 25 MHz. Dieser ist bei Verwendung von eSGM für eine VGA-Auflösung und 25 B/s ausreichend. Aufgrund der Abhängigkeit der Pfadkosten vom Minimum der Vorgänger ist dieser System- und Pixeltakt nur in der eSGM*-Variante realisierbar. Das Minimum der Vorgängerrichtung A wird daher durch eines der drei anderen approximiert und steht somit sofort zu Verfügung. Die Minima der Richtungen B, C und D werden weiterhin in einer Pipelinestruktur berechnet und erst mit jedem Vorschub einer neuen Zeile benötigt.

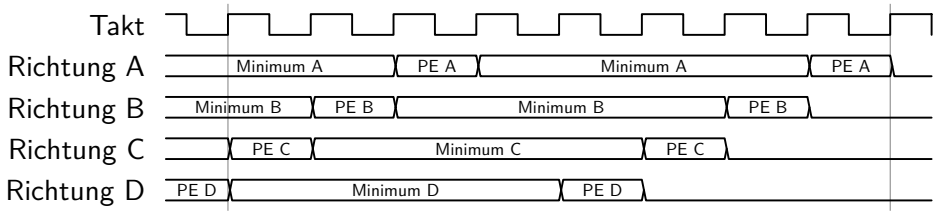


Tabelle 9.1.: Timing-Diagramm für die Berechnung der Pfadkosten in einer Pipeline-Struktur

Eine parallele Variante wie in Abb. 7.6 dargestellt, ermöglicht einen Pixeltakt, der dem Systemtakt entspricht. In diesem Fall können VGA-Bilder mit 100 Hz verarbeitet werden.

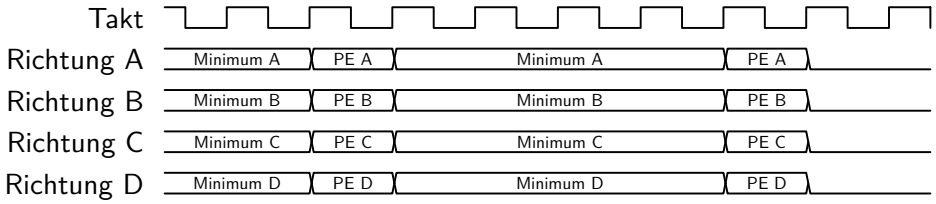


Tabelle 9.2.: Timing-Diagramm für die Pfadkostenberechnung in einer parallelen Struktur

Die Skalierung des Gesamtsystems wird vorrangig von dem Stereo-Matching-Algorithmus bzw. von den Parametern der photogrammetrischen Auswertung bestimmt. Der Messbereich, dessen Genauigkeit und das Zeitverhalten haben einen unterschiedlichen Einfluss auf Skalierbarkeit; bei der Veränderung des Messbereichs ist er beispielsweise quadratisch.

Einfluss der Bildauflösung auf das Laufzeitverhalten

Mit zunehmender Auflösung der Eingangsbilder skalieren sowohl der Ressourcenverbrauch der FPGAs als auch die Rechenzeit linear. Die Veränderung der Auflösung in horizontaler Richtung W bewirkt eine Änderung der FPGA-Logik-Ressourcen und befindet sich innerhalb der Klasse $\mathcal{O}(W)$. Der Mehrbedarf an Ressourcen ist vorrangig der Zwischenspeicherung der vier Pfadkosten der vorhergehenden Zeile geschuldet. Eine Änderung der vertikalen Auflösung H hat einen linearen Einfluss auf die Bearbeitungszeit, so dass die Rechenzeit in der Klasse $\mathcal{O}(W \cdot H)$ liegt.

Einfluss der Anzahl an Disparitätsstufen auf das Laufzeitverhalten

Die Anzahl der Disparitätsstufen hat einen wesentlich höheren Einfluss auf den Ressourcenverbrauch des FPGAs als eine Veränderung der Bildgröße. Die gewählte Realisierung geht von einem parallelen Ansatz aus. Jedes Element einer Pfadrichtung wird zu einem Taktzeitpunkt parallel berechnet.

Die Minimumsuche ist in der Form eines binären Baumes implementiert; die Ausführung der Suche ist somit proportional zur Tiefe des Suchbaumes und von der Ordnung $\mathcal{O}(\log_2 D)$. Ein vollständig kombinatorischer Aufbau ist suboptimal bezüglich der realisierbaren Aktualisierungsrate, so dass Register in die Baumstruktur eingefügt werden müssen. Die Taktrate, mit der die Minimumsuche durchgeführt werden kann, wird so erhöht. Die Latenz erhöht sich entsprechend der Baumtiefe und hat einen Einfluss wie in Abschnitt 8.3 dargestellt.

9.3. Messgenauigkeit

Die Messgenauigkeit des SGM-Algorithmus geht aus der Abb. A.-4 hervor. Die Table 9.3 zeigt einen Ausschnitt, in dem die Unterschiede der verschiedenen SGM-Varianten hinsichtlich der Genauigkeit hervor gehoben sind. Dabei ist zu beachten, dass die Fehlerhäufigkeiten üblicherweise anhand nachbearbeiteter Tiefenkarten bestimmt werden. Fehlzuweisungen werden zum einen durch die Anwendung des L/R-Checks und durch Elimination kleiner Artefakte entfernt. Anschließend werden die betreffenden Fehlstellen durch Interpolation neu bestimmt, wodurch eine weitgehend geschlossene Tiefenkarte entsteht.

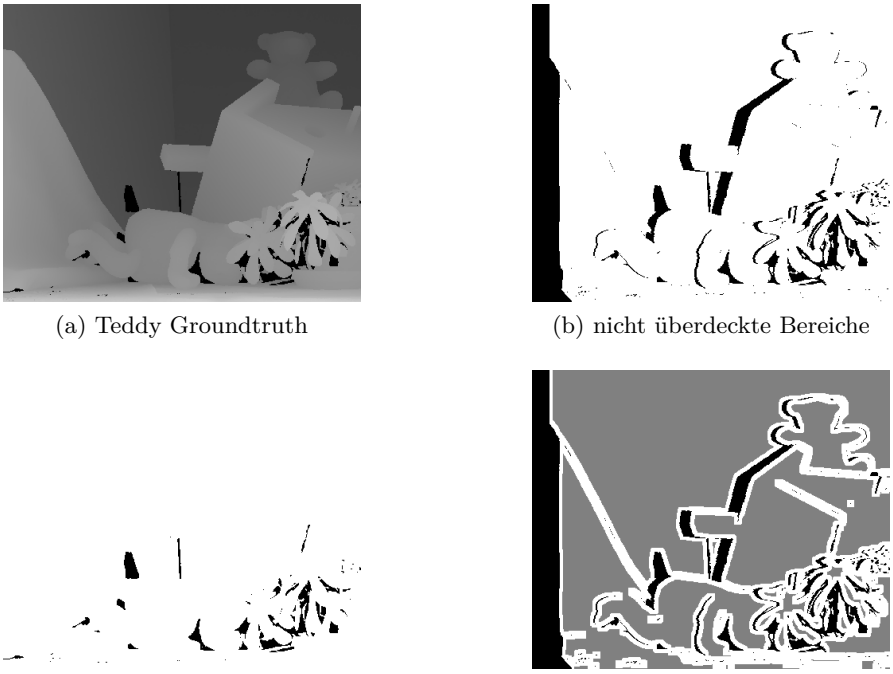
Die Ergebnisse der Version SGM (HMI) basieren auf einer Implementierung, die 2005 von H. Hirschmüller [50] vorgestellt wurde. Die Kostenwerte von cSGM wurden mit Hilfe der Mutual-Information-Methode erstellt und beinhalten eine verbesserte Nachbearbeitung der Disparitätskarte [51]. Die SGM (Census)-Variante entstand nachträglich und ist als Ergänzung bzw. Erweiterung von [51] bezüglich der Kostenfunktion zu sehen.

Algorithmus	Fehler in nicht verdeckten Bereichen				Durchschnittlicher Fehler†
	Tsukuba	Venus	Teddy	Cones	
cSGM [51]	2.61	0.25	5.14	2.77	2.7
SGM (HMI) [50]	3.26	1.00	6.02	3.06	3.3
eSGM (Census 5×5)	3.57	0.72	5.38	3.17	3.2
SGM (Census 5×5)	3.49	0.83	7.35	3.32	3.7
eSGM* (Census 5×5)	3.78	0.69	7.09	3.23	3.7

Tabelle 9.3.: Messergebnisse der SGM-Varianten mit Nachbearbeitung der Disparitätskarten († = in nicht verdeckten Bereichen und > 1)

Als Vergleichsbasis werden die Bilder Tsukuba (Abb. A.21), Venus (Abb. A.23), Teddy (Abb. A.25) und Cones (Abb. A.27) aus section A.3 verwendet. Die Auflösung von Teddy und Cones beträgt 450×375 Pixel und 60 Disparitätsstufen. Die verwendete Auflösung für Venus und Tsukuba beträgt 434×383 bzw. 384×288 Pixel. Der Suchbereich insbesondere für Tsukuba kann auf 5 bis 16 Disparitätsstufen eingeschränkt werden. Das Venus-Bild hat maximal 20 Disparitätsstufen. Der durchschnittliche Fehler ist ein Maß für die Fehlzuweisungen in den drei Kategorien „Randregion bei Tiefensprüngen“ 9.2d, den „nicht verdeckten Bereichen“ 9.2b und zuletzt den „nicht oder halbüberdeckten Bereichen“ 9.2c eines Testbildes. Insbesondere die Wertung der halbüberdeckten Bereiche wird stark von den Nachbearbeitungsschritten beeinflusst.

Die Messergebnisse für die Algorithmen eSGM, eSGM* und SGM (Census 5×5) wurden durch eine von [53] abweichende Nachbearbeitung erstellt. In H. Hirschmüller et al. [53] werden Fehlzuweisungen, kleine Patches, Überdeckungen, Intensitätsüber-



(a) Teddy Groundtruth

(b) nicht überdeckte Bereiche

(c) nicht überdeckte Bereiche + halbüberdeckt

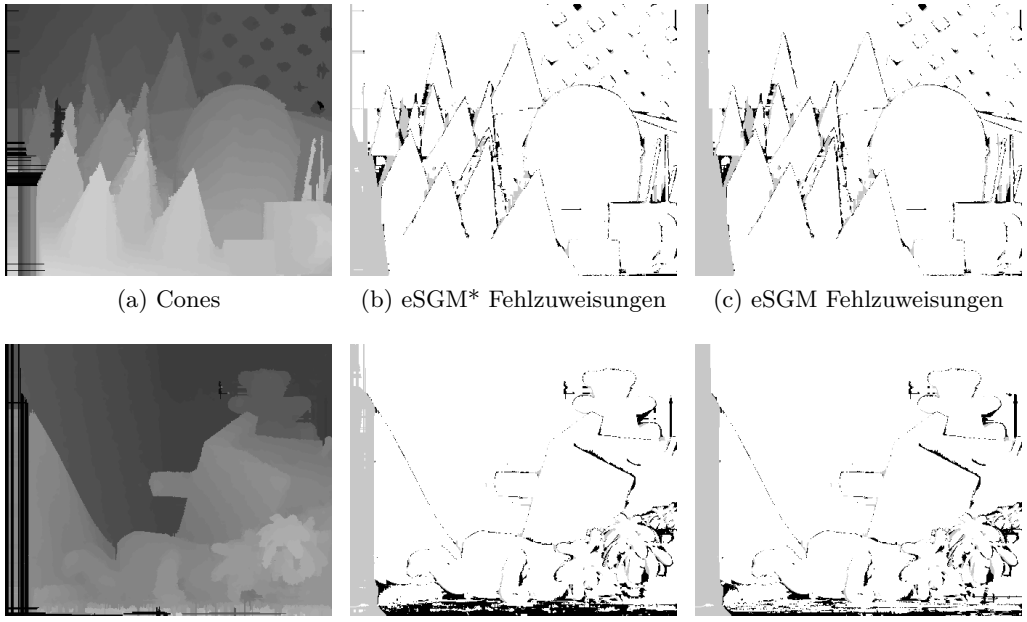
(d) Randzonen

Abbildung 9.2.: Untersuchte Bereiche am Beispiel des Teddy-Bildpaares [97]; nur weiße Regionen werden gewertet

gänge, Disparitätssprünge und schwach texturierte Bereiche gesondert betrachtet, wodurch die durchschnittliche Fehleranzahl in den Regionen für nicht überdeckte Bereiche, halbüberdeckte und Randgebiete weiter minimiert werden kann. In den eSGM- und eSGM*-Algorithmen werden lediglich ungültige Stellen, die durch Eliminierung kleiner Regionen, Überdeckungen und Fehlzuweisungen entstanden sind, durch eine einfache segment-basierte Interpolation, wie sie in [52] beschrieben wurde, entfernt. Überdeckungen und Fehlzuweisungen werden nicht gesondert betrachtet, so dass gerade die Interpolation in halbüberdeckten Bereichen suboptimal ist. Die Beschränkung auf weniger komplexe Interpolations- bzw. Nachbearbeitungsschritte ist durch die hohen Anforderungen an die Berechnungsdauer begründet.

Zur besseren Vergleichbarkeit der hier vorgestellten Realisierungen ist es zweckmäßiger, die Ergebnisse der ursprünglichen SGM-Variante neu zu berechnen und sie anschließend mit den Werten des neuen eSGM-Algorithmus zu vergleichen. Anhand Table 9.3 lässt sich sehr gut die Leistungsfähigkeit von eSGM und eSGM* in Bezug zu der Ausgangslage bewerten.

Der direkte Vergleich von eSGM und eSGM* bestätigt die Entscheidung zugunsten einer effizienteren Berechnung der Disparitätskarte. Am häufigsten treten Unterschiede an den Rändern auf, die durch die allgemein geringe Stützung der Randwerte bedingt



(a) Cones (b) eSGM* Fehlzuweisungen (c) eSGM Fehlzuweisungen

(d) Teddy (e) eSGM* Fehlzuweisungen (f) eSGM Fehlzuweisungen

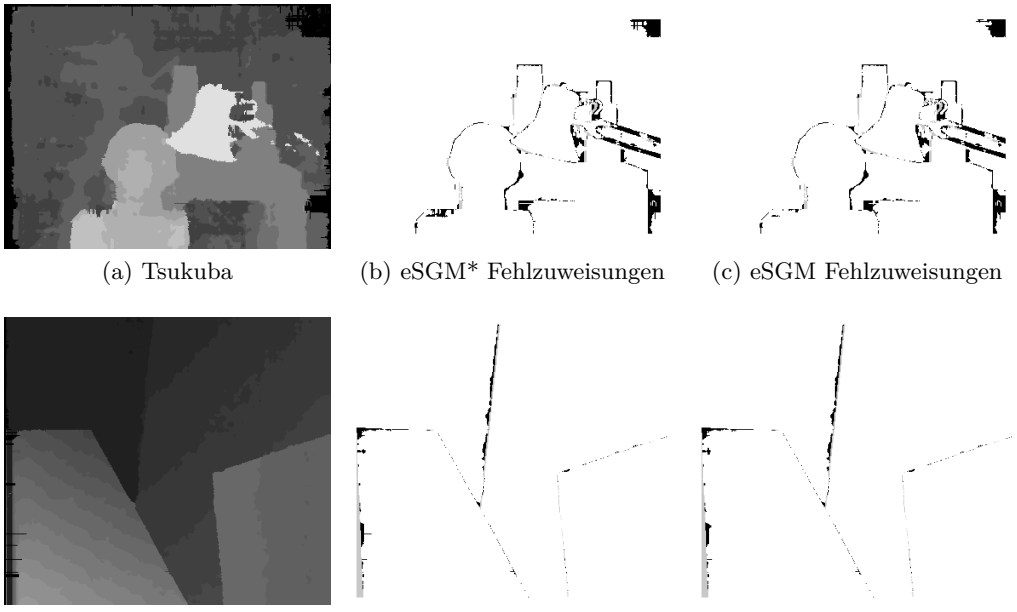
Abbildung 9.3.: eSGM* Ergebnisse im Vergleich zu eSGM. Schwarz markierte Bereiche sind Fehler. Grau markierte Bereiche werden nicht betrachtet. Weiß stellt korrekte Zuordnungen dar.

sind und bei den nicht frontal-parallelen Objekten durch Interpolation korrigiert werden können. Im Vordergrund von Abb. 9.3e ist dies im Vergleich zu der eSGM-Form aus Abb. 9.3f besonders deutlich zu erkennen.

Einfluss der Fenstergröße auf die Messgenauigkeit

Abb. 9.5 verdeutlicht den Einfluss der Fenstergröße bei der Bestimmung des Kostenmaßes. Die prozentual gesehen meisten Fehlzuweisungen für die ersten vier Testbilder aus section A.3 entstehen erwartungsgemäß bei Fenstergrößen, bei denen mindestens eine Dimension relativ klein gewählt wurde.

Insbesondere Fenster der Größe 3×3 Pixel sind ungeeignet. Aufgrund der relativ einfachen Berechnungsvorschrift für das Census-Kostenmaß ist der Mehraufwand für die Erweiterung auf ein 5×5 Pixel großes Fenster gerechtfertigt. Hintergrund ist die bessere Charakterisierung eines Pixels durch die größere Anzahl der durch Census codierten Zustände. Aus Abb. 9.5 wird zudem ersichtlich, dass ab 21 Zuständen, d. h. zum Beispiel einem Fenster der Größe 3×7, keine wesentliche Verbesserung der Korrespondenzanalyse durch SGM bzw. eSGM zu erwarten ist. Die Ursache liegt im wesentlichen darin begründet, dass der SGM-Algorithmus nicht von der verbesserten Kostenfunktion profitieren kann.



(a) Tsukuba (b) eSGM* Fehlzuweisungen (c) eSGM Fehlzuweisungen

(d) Venus (e) eSGM* Fehlzuweisungen (f) eSGM Fehlzuweisungen

Abbildung 9.4.: eSGM* Ergebnisse im Vergleich zu eSGM. Schwarz markierte Bereiche sind Fehler. Grau markierte Bereiche werden nicht betrachtet. Weiß stellt korrekte Zuordnungen dar.

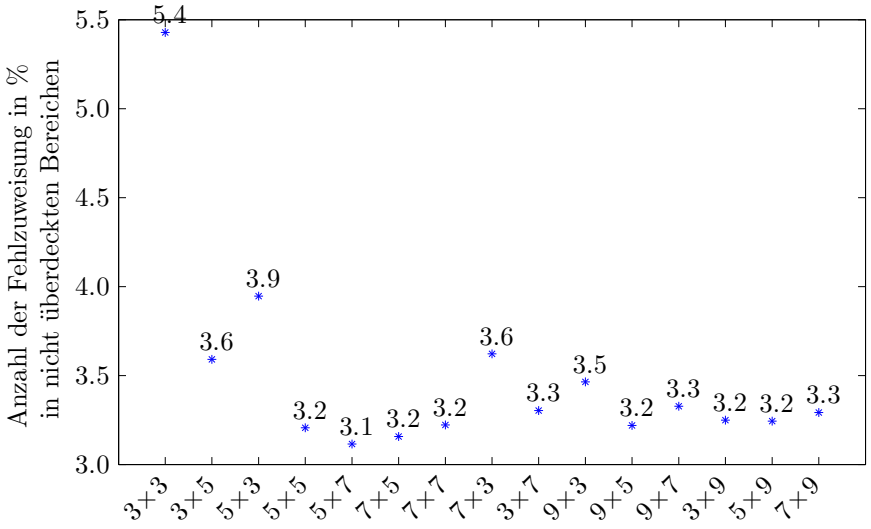


Abbildung 9.5.: Durchschnittlicher Fehler für unterschiedliche Fenstergrößen in nicht verdeckten Bereichen. Alle Angaben in Prozent.

9.4. Diskussion

Die Übersicht und die Referenzdatensätze von D. Scharstein et al. [97] sind hilfreich, eine qualitative Aussage bezüglich der erstellten Disparitätskarten zu treffen und damit ein Mindestmaß an Vergleichbarkeit zwischen den Verfahren herzustellen. Die Testdatensätze spiegeln jedoch nur zum Teil die charakteristischen Eigenschaften realer Messungen wider. Problematisch ist weiterhin, dass neu vorgestellte Algorithmen häufig auf diese Datensätze hin optimiert wurden. Abgesehen davon, kann anhand der Referenz [97] keine Aussage über die Komplexität und die damit einhergehende Laufzeit der verglichenen Verfahren getroffen werden.

Unterschiede zwischen realen und Laborbedingungen zeigen sich insbesondere anhand des Bildes Abb. A.21, das frontal-parallele Modelle mit sehr geringer Disparität bevorzugt. Zusätzlich ist es sehr gut für Farbsegmentierungen geeignet. Des Weiteren wurden die Aufnahmen unter idealen Lichtverhältnissen erstellt und sind dementsprechend relativ rauscharm. Schattierte Bereiche treten ebenfalls nur sehr wenig auf und die Belichtungsdauer ist optimal gewählt. Realistische Aufnahmen hingegen sind im Außenbereich häufig schlecht belichtet. Dies zeigt sich besonders bei schnellem Wechsel zwischen dunklen und hellen Orten wie zum Beispiel bei Tunneldurchfahrten bzw. wechselndem Gegenlicht. Außenaufnahmen zeichnet zudem ein vergleichsweiser hoher Dynamikumfang aus, der erhöhte Anforderungen an den optischen Sensor stellt.

Im Innenraum sind die Anforderungen an das radiometrische Auflösungsvermögen der Kamera zwar geringer, doch ist die schlechtere Ausleuchtung im Innenbereich gleichzeitig die Ursache für ein relativ schlechtes SNR. Nur mit viel Aufwand kann im Innenbereich eine gleichmäßig gute Beleuchtung eingerichtet werden. Im Labor wiederum sind die Lichtverhältnisse von vornherein optimal und über längere Zeit konstant.

Einen entscheidenden Einfluss auf das Laufzeitverhalten bzw. die erforderlichen Datenraten des Semi-Global-Matching-Algorithmus hat die Hardware. Den höchsten Freiheitsgrad hinsichtlich einer optimalen Auslegung des SGM-Algorithmus bieten auf Kundenwunsch spezifizierte Schaltkreise (ASICs), die allerdings mit erheblichen Kosten verbunden sind. Eine Alternative sind FPGAs, die als Zielplattform in dieser Arbeit verwendet werden. Der Entwurf ist hier über zwei gegensätzliche Ansätze möglich. Die Algorithmen können entweder parallel auf die Ressourcen des FPGAs oder sequentiell in die Zeit abgebildet werden. Letzteres hat die sequentielle Abarbeitung von Subroutinen zur Folge, während je nach Umfang der verfügbaren Hardware durchaus auch mehrere parallelisierbare Routinen zur gleichen Zeit berechnet werden können.

Alternative Echtzeit-Stereo-Matching-Verfahren aus Table 5.1 mit vergleichbar guten Messwerten sind in der Regel komplexer. Um dennoch eine adäquate Auflösung und Bildwiederholrate zu erreichen, werden in erster Linie leistungsfähige GPUs verwendet. Das gilt für PlaneFitBP von Q. Yang et al. [117], RealTimeBFV [63] und RealTimeBP von Q. Yang et al. [118], die entweder lediglich ein Bild pro Sekunde oder nur eine sehr

geringe Tiefenauflösung bieten. RealtimeABW [15], FastAggreg [107] und RealTimeVar [66] sind nominal betrachtet mit eSGM* vergleichbar. Zudem werden die Laufzeiten für Standard-CPUs angegeben, so dass Potential für eine Beschleunigung vorhanden wäre. Jedoch wird aus Table 9.4 insbesondere für die Algorithmen RealtimeABW und FastAggreg [107] die Präferierung frontal-paralleler Flächen deutlich. Dies ist eine Eigenschaft, die in realen Umgebungen zu sehr ungenügenden Ergebnissen führt und sich bereits durch die relativ schlechten Werte für Teddy und Cones zeigt. RealTimeVar [66] ist aufgrund seiner numerischen Komplexität für FPGA-Implementierungen nicht geeignet.

Algorithmus	Fehler in nicht verdeckten Bereichen				Durchschnittlicher Fehler
	Tsukuba	Venus	Teddy	Cones	
cSGM [48]	2.61	0.25	5.14	2.77	2.96
PlaneFitBP [117]	0.97	0.17	6.65	4.17	2.99
eSGM (Census 5×5)	3.57	0.72	5.38	3.17	3.21
SGM (HMI) [48]	3.26	1.00	6.02	3.06	3.34
eSGM* (Census 5×5)	3.78	0.69	7.09	3.23	3.70
RealtimeVAR [66]	3.33	1.15	6.18	4.66	3.83
RealTimeBP [118]	1.49	0.77	8.72	4.61	3.90
RealtimeABW [15]	1.26	0.33	10.7	4.81	4.28
RealtimeGPU [115]	2.05	1.92	7.23	6.41	4.40
RealtimeCensus [58]	5.08	1.58	7.96	4.10	4.68
RealTimeBFV [63]	1.71	0.55	9.90	6.66	4.71
FastAggreg [107]	1.16	4.03	9.04	5.37	4.90
ReliabilityDP [85]	1.36	2.35	9.82	12.9	6.61
SSD [97]	5.23	3.74	16.5	10.6	9.02

Tabelle 9.4.: Prozentualer Fehler in nicht verdeckten Bereichen von eSGM und SGM gegenüber alternativen Stereo-Matchern (Fehler > 1 Disparität)

RealtimeCensus [58], RealtimeGPU [115] und ReliablityDB [85] schneiden trotz des starken Hardwareeinsatzes durch GPUs im Vergleich schlechter ab. Die SSD-Implementierung von D. Scharstein et al. [97] dient zu Veranschaulichung der Leistungsfähigkeit lokaler Stereo-Matcher, wie sie in der Industrie und Forschung verwendet werden (Table 5.2 und Table 5.4), die erwartungsgemäß um ein Vielfaches schlechter ist.

Weiterhin erwähnenswert ist die nur in Teilen veröffentlichte Arbeit der Daimler AG, Stuttgart [31]. Wie in section 5.3 geschildert, basiert deren Entwicklung ebenfalls auf dem SGM-Algorithmus und ist für den Einsatz in Fahrzeugen bestimmt. Aufgrund fehlender Informationen kann diese Realisierung nicht in Table 9.4 aufgenommen werden. Die Verwendung des SGM-Verfahrens unterstreicht jedoch sein Potential.

10. Anwendungsbeispiel

In vielen Anwendungen werden Tiefeninformationen unterstützend eingesetzt. In diesem Kapitel wird ein Anwendungsfall vorgestellt, in dem die Generierung von Tiefeninformation aus Stereokamerasystemen zur Unterstützung auf dem Gebiet der Innenraumnavigation [39] Verwendung findet und in Kombination mit der Erstellung von 3D-Modellen des Innenraums notwendig ist.

Die hochgenaue Bestimmung der eigenen Lage ist eine Grundvoraussetzung für viele Anwendungen. Ein häufig zitiertes Beispiel ist die Bestimmung der Position und Orientierung des eigenen Fahrzeugs im Straßenverkehr bzw. auf der Landkarte. Die Mehrzahl der verfügbaren Verfahren und Algorithmen ist auf eine Form von technischer Infrastruktur wie zum Beispiel GPS angewiesen. In Gebieten mit nicht ausreichendem GPS-Empfang werden typischerweise zusätzliche Hilfsmittel im Vorfeld installiert. Dazu zählen sowohl aktive Systeme, wie zum Beispiel Ergänzungssysteme für GPS zur Steigerung der Genauigkeit oder Funk-Sender zur Peilung und WLAN-Netze im Innenbereich, als auch RFID-Tags und jede sonstige Art von passiven Markierungen. Inertiale Messsysteme (IMUs) für den kommerziellen bzw. nicht-militärischen Bereich können diese Lücke sowohl aus praktischen als auch aus Kostengründen alleine nicht füllen. Im Innenraum wird die Situation dadurch verschärft, dass von vornherein keine absolute Referenzierung durch das GPS möglich ist.

Das Projekt Integrales-Positioning-System (IPS) [39] ist ein DLR-Vorhaben zur Entwicklung eines Multisensorsystems zur Bestimmung der eigenen Lage sowohl im Innen- als auch im Außenbereich, das ohne jegliche zusätzliche Infrastruktur auskommt. Die Hauptsensoren bestehen aus einem Kamerapaar, das Stereoaufnahmen von der Umgebung erstellt. Neben der Auswertung der Stereobilder werden Messdaten von Beschleunigungssensoren gewonnen.

Mit dem Stereosystem wird die relative Lageänderung des Nutzers gemessen, indem von einem Aufnahmezeitpunkt zum nächsten die Verschiebung gefundener Features im Bild gemessen wird. In Abb. 10.1 sind das linke und das rechte Kamerabild zu erkennen. Die roten Kreuze stellen gefundene Features dar, die zur Triangulation verwendet werden. Durch schnelle Bewegungen, hohe Belichtungsdauern (Bewegungsunschärfe) bzw. das generelle Fehlen von Features kann die Navigation mit Hilfe von Kameras ausfallen. Im Fall eines Nutzers, der den Sensorkopf in der Hand trägt und einen Flur wie in Abb. 10.1 dargestellt entlangläuft, bewirken bereits Rotationsbewegungen des Stereokopfes von moderaten 50 °/s, die einer sanften Handbewegung entsprechen, erhebliche Änderungen im Bild. Zur Veranschaulichung: Angenommen, der IFOV eines

Abbildung 10.1.: Rohbilder, die vom IP-System aufgenommen wurden. Ort: DLR-Standort Oberpfaffenhofen / Wessling, RM-Gebäude, Flur EG

Pixels beträgt 0.1 ° und das Aufnahmesystem arbeitet mit 10 Hz, dann resultiert daraus eine Pixelbewegung von 50 Pixeln im Radius in zwei aufeinanderfolgenden Bildern. Für echtzeitkritische Anwendungen ist die Einschränkung des Suchbereichs kritisch, sodass Zusatzinformation von IMUs hilfreich ist.

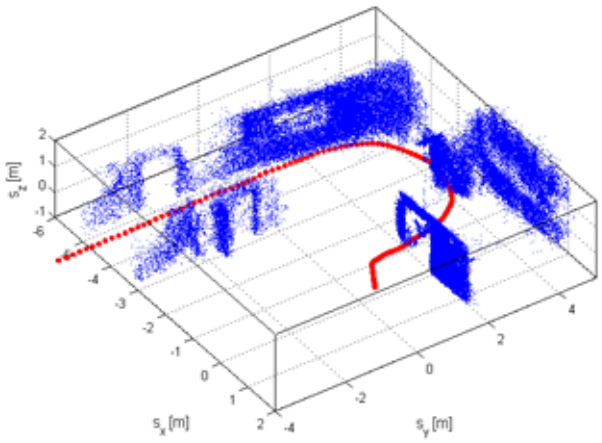


Abbildung 10.2.: Die Tiefenmessungen (blau dargestellt) auf Grundlage der Stereobildauswertung wurden mit den Navigationsdaten des IP-Systems referenziert. Rot eingezeichnet ist die Trajektorie des Messsystems.

Im Gegensatz zu den Kameras erfassen IMUs zu jeder Zeit ihre Messgrößen unabhängig von der Umgebung. IMUs sind mit Gyroskopen und Beschleunigungssensoren ausgestattet, mit deren Hilfe relative Lageänderungen erfasst werden können. Da mit IMUs die Rotation und Translation nur relativ erfasst werden können, kommt es zur Aufsummierung der Messfehler in den Sensordaten einer IMU. Optische Systeme messen Rotations- und Translationsänderungen, sofern homologe Punkte existieren, direkt und

sind diesbezüglich frei von systematischen Fehlern. Aus diesem Grund wird die Bewegungsinformation aus der Stereobildauswertung zur Eingrenzung der Inertialsensoren herangezogen. Die IMU wiederum wird gebraucht, um den Suchbereich bei der Featuredetektion einzugrenzen. Das IP-System verfolgt damit den Ansatz, die spezifischen Nachteile der jeweiligen Sensoren durch die Stärken des anderen zu kompensieren. Es entsteht eine zum einen robuste und ausfallsichere (redundante) Navigationslösung, die zum anderen eine relativ hohe Güte hat und in Echtzeit eine Lösung liefert.

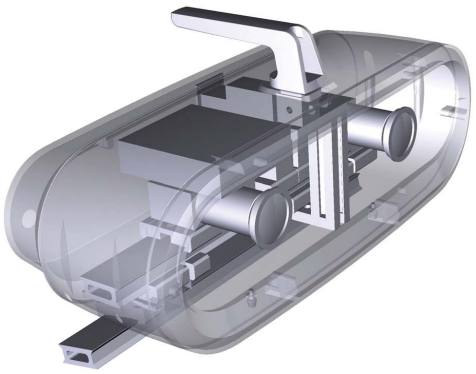


Abbildung 10.3.: IPS-Kamerakopf für den Innenraum-Einsatz.

Durch die bestimmten relativen Orientierungs- und Positionsänderungen des Kamerakopfes (Abb. 10.3) ist es im Anschluss möglich, diese Information für die geometrische Referenzierung der Tiefeninformation zu verwenden. Die Tiefeninformation wird parallel zur Navigationslösung berechnet. Somit lassen sich die 3D-Punktwolken zu jedem Stereoaufnahmezeitpunkt räumlich zueinander in Bezug bringen, wodurch wiederum ein 3D-Modell der Umgebung erzeugt werden kann.

Sowohl die Eingangsbilder für das Echtzeit-Stereo-Matching als auch für das IPS wurden zur selben Zeit mit dem gleichen Kamerakopf aus Abb. 10.3 erstellt. Für das im Innenraumbereich konzipierte Messsystem kommen zwei Monochrom-Kameras zum Einsatz. Weiterhin sind eine Inertial-Messeinheit und ein Neigungssensor verbaut. Dabei handelt es sich im Fall von Abb. 10.1 um eine Messkampagne im Innenraum des Instituts für Robotik- und Mechatronik in Oberpfaffenhofen/Wessling. In Abb. 10.2 ist rot gekennzeichnet die Trajektorie des Messsystems zu erkennen, wie sie durch das IP-System bestimmt wurde. Die blauen Punkte stellen die gewonnene Tiefeninformation dar, die bereits geometrisch referenziert wurden. Zu erkennen sind demnach die Wände von Innenräumen. In der Abbildung 10.2 sind ebenfalls sehr gut die Türen und Durchgänge zu erkennen.

In einem nächsten Schritt können die fusionierten Daten für Mapping-Aufgaben genutzt werden (Abb. 10.4), wodurch die Navigation erheblich verbessert werden kann. Weiterhin ist es denkbar, die dichte Tiefeninformation direkt in die Navigationslösung

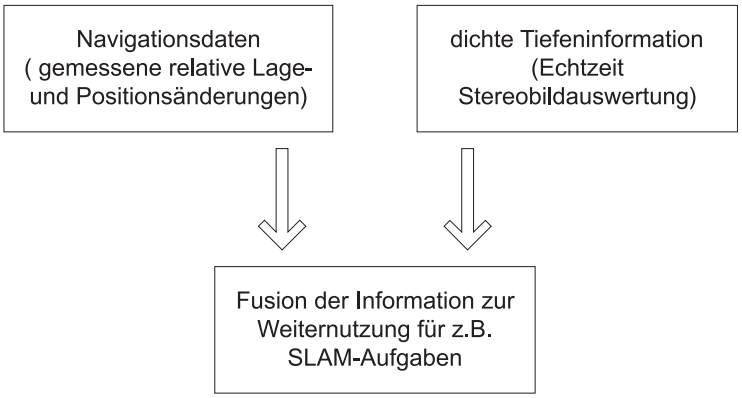


Abbildung 10.4.: Weiterführende Nutzung der geometrisch registrierten Tiefeninformation.

einfließen zu lassen. Für Punkte im Stereobild, deren Disparitätswert Null ist, können in zeitlich aufeinanderfolgenden Bildsequenzen Translationsbewegungen ausgeschlossen werden.

11. Zusammenfassung und Ausblick

Diese Arbeit befasst sich mit der Entwicklung eines echtzeitfähigen Systems zur Erstellung von Tiefeninformation aus Stereobildpaaren, das in einer Reihe von Anwendungen zur dreidimensionalen Vermessung des Raumes herangezogen werden kann. Als Hauptanwendungsgebiete sind in erster Linie mobile Robotikapplikationen vorgesehen, die sehr strenge Anforderungen sowohl bezüglich des Ressourcenverbrauchs als auch im Hinblick auf die Messeigenschaften und das Laufzeitverhalten stellen. Ein Merkmal des in dieser Arbeit entworfenen Systems ist die in Echtzeit stattfindende Ausführung der verwendeten Algorithmen in Kombination mit sehr guten Messeigenschaften, so dass es möglich ist, die in section 6.1 genannten allgemeinen und in section 7.1 spezifizierten konkreten Anforderungen zu erfüllen. Im Einzelnen bedeutet dies, dass durch das System 25 Bilder in der Sekunde in VGA-Auflösung verarbeitet und als Disparitätskarten mit 128 Stufen dem Nutzer zur Verfügung gestellt werden können. Das verwendete Stereo-Matching-Verfahren basiert auf einem globalen Ansatz und liefert im Vergleich zu den alternativen echtzeitfähigen Methoden sehr gute Ergebnisse, die in chapter 9 vorgestellt werden.

Die notwendigen theoretischen Voraussetzungen für diese Arbeit werden in chapter 2 erläutert, in dem unter anderem die Bildgewinnung und -vorverarbeitung sowie die Grundlagen der Stereobildauswertung behandelt werden. Die verschiedenen Verarbeitungsarchitekturen und -paradigmen werden in chapter 4 vorgestellt und leiten über in die Darstellung des verwendeten Entwurfsprozesses für das heterogene System.

Das Konzept für das System wird in chapter 6 entworfen und anhand allgemeiner Anforderungen ausgelegt. Insbesondere werden die Algorithmen für die Stereo-Bildauswertung vorgestellt. Im Vordergrund steht der Semi-Global-Matching-Algorithmus (SGM). Sowohl im praktischen Außeneinsatz als auch im Vergleich mit den veröffentlichten Ergebnissen, die die Bilder aus [97] zur Grundlage haben, erzielt der Semi-Global-Matching-Algorithmus sehr gute Ergebnisse.

Der ursprüngliche Ansatz des Semi-Global-Matching-Verfahrens besteht darin, ein NP-vollständiges Problem, das für zweidimensionale Bilder definiert ist, durch Lösung einzelner eindimensionaler Teilprobleme zu approximieren. Ein gravierender Nachteil des Algorithmus ist der sehr hohe Ressourcenverbrauch in Bezug auf die Speicherbandbreite und -kapazität. Dies führt dazu, dass der Algorithmus für Echtzeitanwendungen ungeeignet ist. Das Gleiche gilt für die hochwertigen alternativen Matchingverfahren, die mitunter algorithmisch noch weitaus komplexer sind, so dass in Echtzeitapplikatio-

nen nur lokale Stereo-Verfahren Verwendung finden. Lokale Verfahren liefern jedoch im Vergleich zu den globalen Methoden qualitativ schlechte Disparitätskarten.

Der neue Algorithmus Efficient-Semi-Global-Matching (eSGM) ist eine speichereffiziente Form des ursprünglichen Verfahrens. Zusätzlich liefert das eSGM-Verfahren auch bessere Teillösungen für jede eindimensionale Subaufgabe, wodurch insgesamt eine bessere globale Annäherung an das Ausgangsproblem möglich ist. Im qualitativen Vergleich [97] liefert eSGM daher sogar bessere Ergebnisse als der Basisalgorithmus SGM (Table 9.3). Mit dem neuen Verfahren können somit zwei grundsätzlich gegensätzliche Entwicklungsziele, nämlich Senkung des Ressourcenverbrauchs und Verbesserung der Disparitätskarte, erreicht werden.

Das Konzept wird für mobile Robotikanwendungen in chapter 7 umgesetzt. Wegen der begrenzten Ressourcen der realen Hardware wurde mit eSGM* eine Weiterentwicklung des eSGM-Algorithmus für die Realisierung genutzt. Durch die Veränderung sinkt die Qualität der Disparitätskarte leicht auf das Niveau der Basismethode SGM.

In chapter 9 wird das System anhand der drei Kerneigenschaften Laufzeit, Ressourcenverbrauch und Qualität der Tiefeninformation gegenüber den Verfahren nach dem Stand der Technik bewertet. Dabei zeigt sich, dass eine Vergleichbarkeit der Systeme nur teilweise gegeben ist. Insbesondere die Messeigenschaften lassen sich nur unter bestimmten Voraussetzungen vergleichen, denn ein grundsätzliches Problem der Stereo-Bildverarbeitung ist der Mangel an exakten Referenzdaten.

Der in dieser Arbeit vorgestellte FPGA-Ansatz, die eingesetzte Entwurfsmethode und die vorgestellten Algorithmen ermöglichten es, ein leistungsfähiges Stereo-Bildverarbeitungssystem zu entwickeln, das den hohen Anforderungen bezüglich des Laufzeitverhaltens und der Qualität des Ergebnisses gerecht wird.

Im Zuge dieser Arbeit sind eine Reihe neuer Fragestellungen aufgetreten, die im Rahmen zukünftiger Arbeiten noch eingehender betrachtet werden sollten. Insbesondere die Weiterentwicklung der Algorithmen für das Stereo-Matching bietet genug Potential, um eine Steigerung der Ausführungsgeschwindigkeit und der Messgenauigkeit zu ermöglichen. Durch die Einführung eines Konfidenzmaßes auf Basis weiterer Bildauswertung ist es möglich, eine Wichtung der Pfadrichtungen einzuführen. Im Rahmen dieser Arbeit hat sich gezeigt, dass eine Gleichgewichtung der Pfadrichtungen von Nachteil bezüglich der Matchingergebnisse ist. Weiterhin muss der nicht zu vernachlässigende Ressourcenverbrauch des vorgestellten Systems optimiert werden. Beispielsweise bietet sich die regelmäßige Datenstruktur der Richtungsterme

$$L_r(i, j, d)$$

aus Equation 8.1 und folgend für eine ressourcenschonende Codierung an. Konzepte wie von C. Banz et al. [7] sind ebenfalls geeignet, den Bedarf für die Zwischenspeicherung der Richtungsterme zu senken, steigern jedoch den Ressourcenverbrauch erheblich.

Ein weiterer und interessanter Aspekt ist die stärkere Integration zum einen von re-

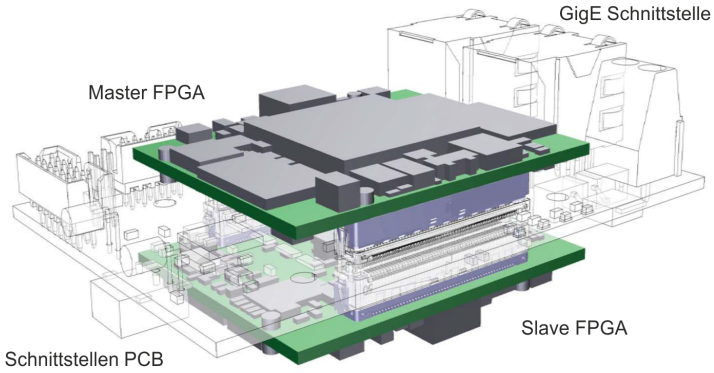


Abbildung 11.1.: Kompakte Bauform des Systems zur Gewinnung der Tiefeninformation für mobile Anwendungen. Die Abbildung ist ein Modell der aktuellen Realisierung.

konfigurierbarer Hardware und zum anderen von Graphikprozessoren. Vorstellbar ist die Entwicklung von Graphikkarten für den wissenschaftlichen Einsatz, die zusätzlich mit FPGAs bestückt sind und über eine gemeinsame Schnittstelle untereinander bzw. zum Hostsystem verfügen. Einschränkend wirkt in diesem Zusammenhang ebenfalls die Anforderung an ein mobiles und energiesparendes System.

In naher Zukunft steht die Operationalisierung des hier gezeigten Systems im Vordergrund. Insbesondere für die mobilen Anwendungsszenarien ist eine stärkere Integration notwendig und wird durch die Realisierung des in Abb. 11.1 gezeigten Modells angestrebt. Eine wichtige Voraussetzung für weitere Arbeiten im Umfeld der Hardwareprogrammierung ist jedoch die Verbesserung der Entwurfsmethoden. Der relativ komplexe Entwurfsprozess und die oftmals fehlenden Schnittstellen werden nach wie vor als Hauptursache für Verzögerungen in der Entwicklung gesehen. Standardisierungen in Form des hier verwendeten Hardwarebetriebssystems bilden die Basis für den Entwurf in dieser Arbeit und müssen bezüglich der Themen Verifikation und Test weiterentwickelt werden.

A. Anhang

A.1. Beispiel Projektkonfiguration

```
1
SOURCE FILES     =    ./top_sgm_top.vhdl,
                      ${SGM_SRC}/sgm_top.vhdl,
                      ${SGM_SRC}/sgm_input.vhdl,
                      ${SGM_SRC}/sgm_s.vhdl,
6                     ${SGM_SRC}/sgm_d.vhdl,
                      ${SGM_SRC}/sgm_pe.vhdl,
                      ${SGM_SRC}/sgm_minimum.vhdl,
                      ${SGM_SRC}/sgm_minimum_operator.vhdl,
                      ${SGM_SRC}/sgm_census.vhdl,
11                    ${SGM_SRC}/sgm_census_transform.vhdl,
                      ${SGM_SRC}/sgm_census_correlate.vhdl,
                      ${SGM_SRC}/sgm_filter.vhdl,
                      ${SGM_SRC}/sgm_control.vhdl,
                      ${SGM_SRC}/sgm_s_fusion.vhdl,
16                    ${SGM_SRC}/sgm_memory.vhdl,
                      ${SGM_SRC}/sgm_memory_d.vhdl,
                      ${SGM_SRC}/sgm_con_check.vhdl,
                      ${SGM_SRC}/sgm_penalty.vhdl,
                      ${PACKAGE_DIR}/basic_lib_package.vhdl,
21                    ${PACKAGE_DIR}/basic_package.vhdl,
                      ${SGM_SRC}/sgm_package.vhdl;

TOPLEVEL NAME                =    top_sgm_top;

26 MAIN ENTITY NAME          =    main_entity;
MAIN ARCHITECTURE NAME       =    main_architecture;
MODULE = clockadaption;      // gilt f\"ur alle Projekte \"ubergreifend


31 sim2
{
  OUTPUT FILE          =    sim2/esgm_single_pass_serial.vhd;
   pipe_input          =    sim_pipe_input;
   pipe_output         =    sim_pipe_output;
36 channel_input       =    sim_channel_input;
   channel_output      =    sim_channel_output;
   schannel_output     =    sim_schannel_output;
   clock               =    sim_clock;
   reset               =    sim_reset;
41 memory              =    sim_aes_ddr2_01_memory;
   memory2             =    sim_aes_ddr2_02_memory;
```

```
    time                =   sim_time;
    MODULE              =   stack2;
    MODULE              =   stack;
    MODULE              =   sfifo_sim;
    MODULE              =   afifo_sim;
    MODULE              =   local_filter;
    MODULE              =   pipehandling;
};


a

aes110_usb
{
    OUTPUT FILE         =   aes110_usb/esgm_single_pass_serial.vhd;
    UCF FILE            =   aes110_usb/esgm_single_pass_serial.ucf;
    pipe_input          =   aes_usb_pipe_input;
    pipe_output         =   aes_usb_pipe_output;
    channel_input       =   aes_usb_channel_input;
    channel_output      =   aes_usb_channel_output;
    schannel_output     =   aes_usb_schannel_output;
    clock               =   aes_clock;
    reset               =   aes_reset;
    MODULE              =   sfifo_xc5v;
    MODULE              =   afifo_xc5v;
    time                =   aes_time;
    MODULE              =   stack_burst;
    MODULE              =   stack_burst2;
    MODULE              =   local_filter;
    MODULE              =   chipscope_xc5v;
    MODULE              =   pipehandling;
    MODULE              =   stack;
    memory              =   aes_ddr2_01_memory_fp;
    memory2             =   aes_ddr2_02_memory_fp;

};

aes110_pcie
{
    OUTPUT FILE         =   aes110_pcie/esgm_single_pass_serial.vhd;
    UCF FILE            =   aes110_pcie/esgm_single_pass_serial.ucf;
    pipe_input          =   aes_pcie_pipe_input;
    pipe_output         =   aes_pcie_pipe_output;
    channel_input       =   aes_pcie_channel_input;
    channel_output      =   aes_pcie_channel_output;
    schannel_output     =   aes_pcie_schannel_output;
    clock               =   aes_clock;
    reset               =   aes_reset;
    MODULE              =   sfifo_xc5v;
    MODULE              =   afifo_xc5v;
    time                =   aes_time;
    MODULE              =   stack_burst;
    MODULE              =   stack_burst2;
    MODULE              =   local_filter;
```

```
  MODULE                  =    chipscope_xc5v;
  MODULE                  =    pipehandling;
  MODULE                  =    stack;
  memory                  =    aes_ddr2_01_memory_fp;
  memory2                 =    aes_ddr2_02_memory_fp;

};
```

Listing A.1: Projektkonfiguration

A.2. Berechnung der Matrix Q (Disparity to Depth)

Gültig im linken Bild des rektifizierten Kamerapaares. Gesucht wird der Objektpunkt P im Kamerakoordinatensystem. Gegeben sind der linke Bildpunkt p'_1 und die Disparität d mit $p'_2 = p'_1 - d$.

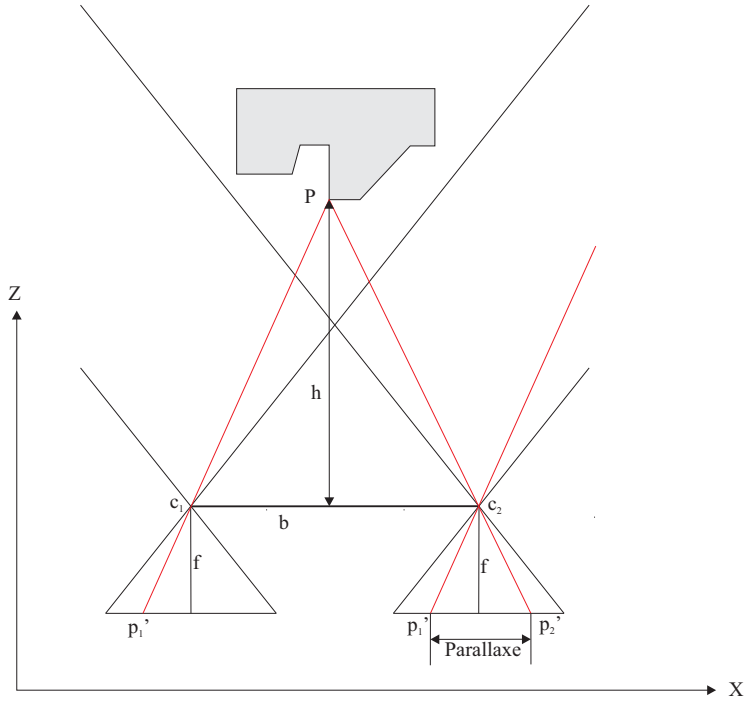


Abbildung A.1.: Der Stereonormalfall [77]

Vorbetrachtung

$$p'_1 = \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{A.1}$$

$$\begin{aligned} m &= b - x - (-x') && \text{(A.2)} \\ &= b - x + x - d && \text{(A.3)} \\ &= b - d && \text{(A.4)} \end{aligned}$$

Durch Anwendung des Strahlensatzes und durch Umformung kann für die Tiefe Z hergeleitet werden:

$$\frac{b}{Z} = \frac{m}{Z-f} \tag{A.5}$$
$$\Longrightarrow \tag{A.6}$$
$$b \cdot (Z-f) = m \cdot Z \tag{A.7}$$
$$= (b-d) \cdot Z \tag{A.8}$$

$$bZ - bf = bZ - dZ \tag{A.9}$$
$$\Longrightarrow \tag{A.10}$$
$$Z = \frac{b \cdot f}{d} \tag{A.11}$$

Weiterhin gilt das Gleichungssystem:

$$\lambda p_1' = P \tag{A.12}$$
$$\lambda \begin{bmatrix} x \\ y \\ f \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{A.13}$$

Durch Ersetzen der dritten Zeile ergibt sich:

$$\lambda f = Z \tag{A.14}$$
$$= \frac{b \cdot f}{d} \tag{A.15}$$
$$\Longrightarrow \tag{A.16}$$
$$\lambda = \frac{b}{d} \tag{A.17}$$

, sodass für den Objektpunkt P gilt:

$$P = \frac{b}{d} \cdot \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{A.18}$$

Für die Transformation der Bildpunkte aus Kamerakoordinaten in Pixelkoordinaten gilt:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & a_{02} \\ 0 & f & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{A.19}$$

Gesucht wird die Matrix Q in homogener Form, sodass ausgehend vom Bildkoordinatensystem auf die Position des Objektpunktes im Kamerakoordinatenmodel geschlossen werden kann, wobei lediglich die Disparität d' in Pixel zweier korrespondierender Punkte bekannt ist. Bildpaare müssen rektifiziert sein.

$$\begin{bmatrix} P \\ 1 \end{bmatrix} = \lambda \cdot Q \cdot \begin{bmatrix} u \\ v \\ d' \\ 1 \end{bmatrix} \tag{A.20}$$

Durch ersetzen der Koordinatensysteme ergibt sich:

$$\begin{bmatrix} P \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{A.21}$$

$$= \begin{bmatrix} \frac{b}{d}x \\ \frac{b}{d}y \\ \frac{b}{d}f \\ 1 \end{bmatrix} \tag{A.22}$$

$$= \begin{bmatrix} \frac{b}{d'}(u - a_{02}) \\ \frac{b}{d'}(v - a_{12}) \\ \frac{b}{d'} \cdot f \\ 1 \end{bmatrix} \tag{A.23}$$

und nach Umstellung:

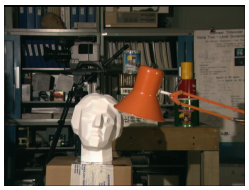
$$\begin{bmatrix} \frac{b}{d'}(u - a_{02}) \\ \frac{b}{d'}(v - a_{12}) \\ \frac{b}{d'} \cdot f \\ 1 \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d' \\ 1 \end{bmatrix} \tag{A.24}$$

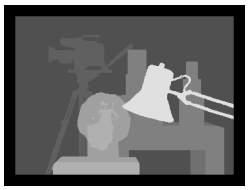
$$= \frac{d'}{b} \cdot Q \begin{bmatrix} u \\ v \\ d' \\ 1 \end{bmatrix} \tag{A.25}$$

mit

$$Q = \begin{bmatrix} 1 & 0 & 0 & -a_{02} \\ 0 & 1 & 0 & -a_{12} \\ 0 & 0 & 0 & a_{00} \\ 0 & 0 & \frac{1}{b} & 0 \end{bmatrix} \tag{A.26}$$

A.3. Referenz-Datensätze

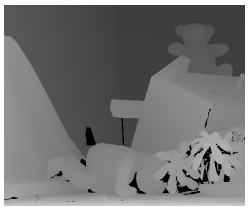

(1) Tsukuba


(2) Groundtruth

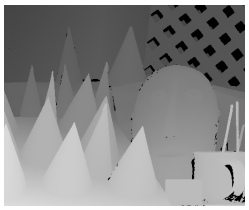
(3) Venus


(4) Groundtruth

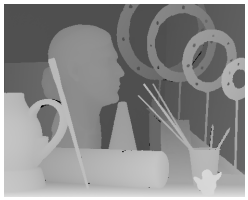

(5) Teddy


(6) Groundtruth

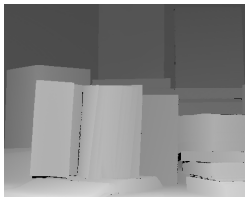

(7) Cones


(8) Groundtruth

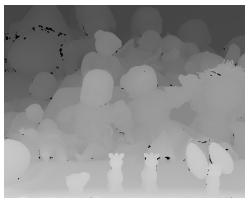
(9) Art


(10) Groundtruth

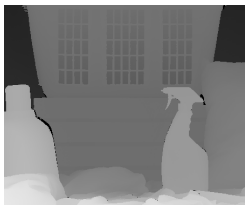
(11) Books


(12) Groundtruth


(13) Dolls


(14) Groundtruth

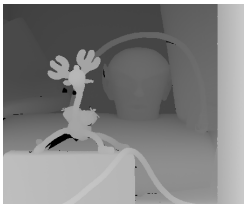

(15) Laundry


(16) Groundtruth

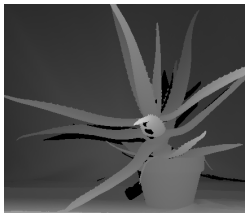

(17) Moebius


(18) Groundtruth

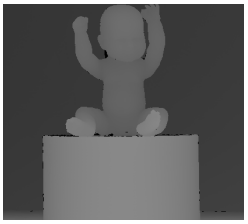

(19) Reindeer

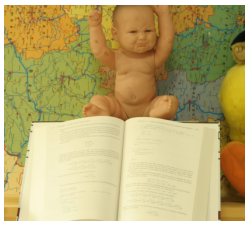

(20) Groundtruth

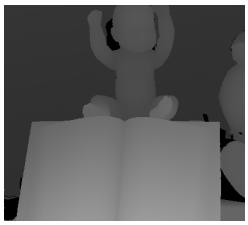

(21) Aloe


(22) Groundtruth

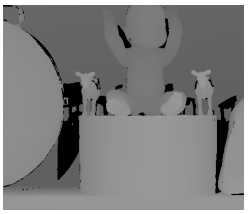

(23) Baby1


(24) Groundtruth

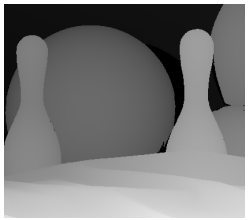

(25) Baby2


(26) Groundtruth

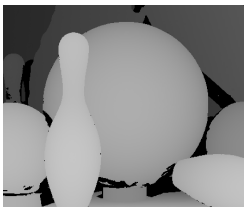

(27) Baby3

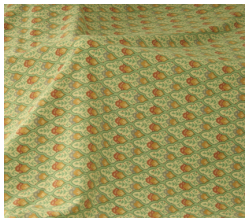

(28) Groundtruth

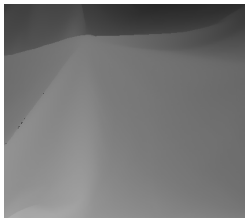

(29) Bowling1


(30) Groundtruth

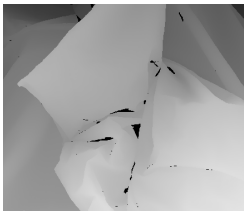

(31) Bowling2


(32) Groundtruth

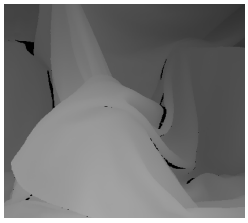

(33) Cloth1


(34) Groundtruth

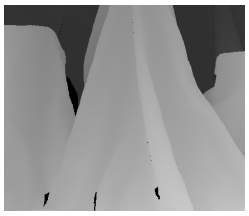

(35) Cloth2


(36) Groundtruth

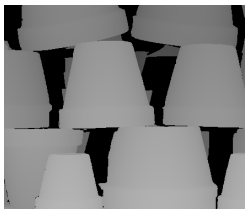

(37) Cloth3


(38) Groundtruth

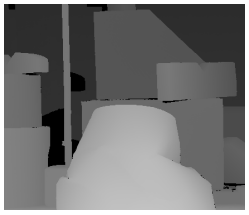

(39) Cloth4


(40) Groundtruth

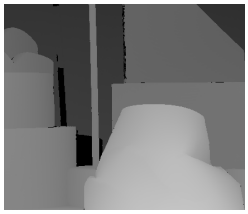

(41) Flowerpots


(42) Groundtruth

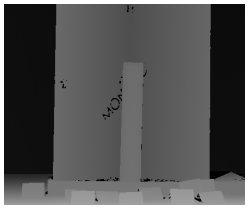

(43) Lampshade1


(44) Groundtruth

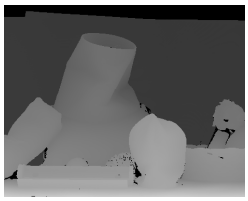

(45) Lampshade2


(46) Groundtruth

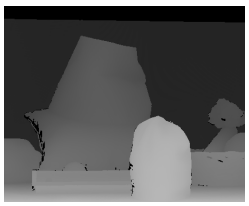

(47) Monopoly


(48) Groundtruth

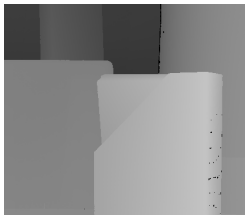

(49) Midd1


(50) Groundtruth

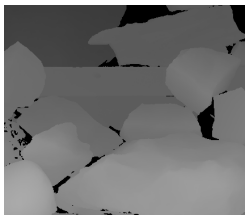

(51) Midd2


(52) Groundtruth

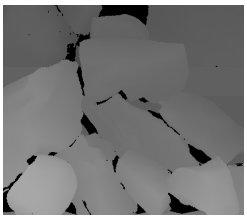

(53) Plastic


(54) Groundtruth


(55) Rocks1

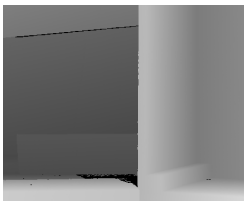

(56) Groundtruth

(57) Rocks2



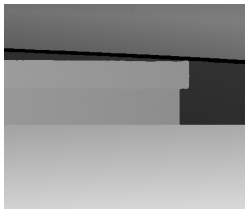
(58) Groundtruth

(59) Wood1



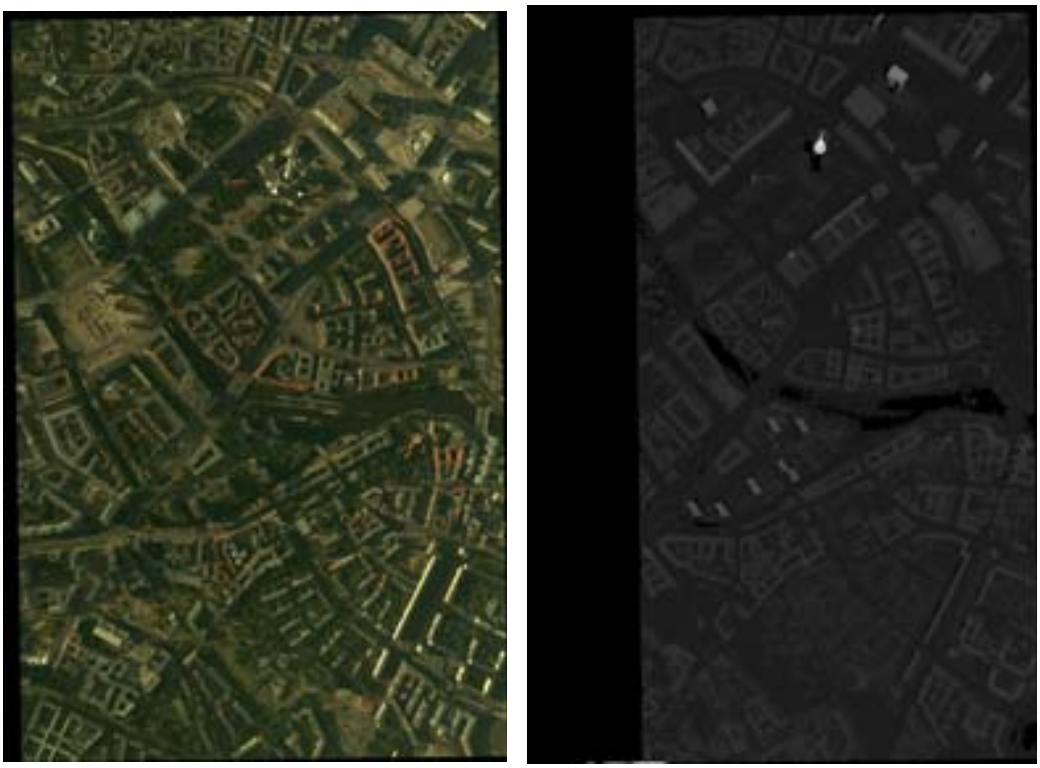
(60) Groundtruth

(61) Wood2



(62) Groundtruth

A.4. Zusätzliches Bildmaterial



(1) Eingangsdaten (2) Tiefenbild

Abbildung A.-5.: Stereobildverarbeitung einer Satellitenaufnahme der Berliner Innenstadt. (Quelle: [53])

Error Threshold = 1 (Error Threshold...)		Sort by nonocc			Sort by all			Sort by disc			
Algorithm	**Avg. Rank**	**Tsukuba** ground truth nonocc	all	disc	**Venus** ground truth nonocc	all	disc	**Teddy** ground truth nonocc	all	disc	**Cones** ground truth nonocc	all	disc	Average percent of bad pixels (**explanation**)
AdaptGCP [137]	6.9	1.03 13	1.29 5	5.60 15	0.10 3	**0.14** 1	1.30 4	4.63 16	6.47 6	12.5 17	**1.81** 1	**5.70** 1	**5.33** 1	3.83
ADCensus [94]	10.3	1.07 17	1.48 14	5.73 20	0.09 2	0.25 8	1.15 3	4.10 10	6.22 4	10.9 9	2.42 13	7.25 10	6.95 14	3.97
AdaptingBP [17]	12.8	1.11 21	1.37 8	5.79 22	0.10 4	0.21 5	1.44 6	4.22 12	7.06 9	11.8 13	2.48 15	7.92 19	7.32 19	4.23
CoopRegion [41]	13.1	0.87 4	**1.16** 1	4.61 4	0.11 5	0.21 4	1.54 8	5.16 24	8.31 14	13.0 21	2.79 28	7.18 9	8.01 35	4.41
RVbased [116]	16.9	0.95 9	1.42 12	4.98 9	0.11 7	0.29 13	**1.07** 1	5.98 33	11.6 45	15.4 41	2.35 10	7.61 11	6.81 12	4.88
DoubleBP [35]	17.2	0.88 6	1.29 4	4.76 7	0.13 9	0.45 32	1.87 15	3.53 7	8.30 13	9.63 6	2.90 35	8.78 45	7.79 28	4.19
RDP [102]	17.2	0.97 10	1.39 10	5.00 10	0.21 30	0.38 21	1.89 16	4.84 17	9.94 25	12.6 18	2.53 17	7.69 13	7.38 20	4.57
OutlierConf [42]	18.6	0.88 5	1.43 13	4.74 6	0.18 20	0.26 11	2.40 30	5.01 20	9.12 20	12.8 20	2.78 27	8.57 36	6.99 15	4.60
SubPixDoubleBP [30]	23.6	1.24 29	1.76 34	5.98 27	0.12 8	0.46 34	1.74 12	3.45 6	8.38 15	10.0 8	2.93 38	8.73 42	7.91 30	4.39
SurfaceStereo [79]	23.6	1.28 35	1.65 23	6.78 44	0.19 22	0.28 12	2.61 42	3.12 4	**5.10** 1	8.65 2	2.89 34	7.95 21	8.26 43	4.06
SubPixSearch [127]	23.8	2.04 82	2.48 71	6.40 37	0.14 13	0.40 25	1.74 12	4.00 9	6.39 5	11.0 11	2.24 7	6.87 7	6.50 7	4.18
LLR [135]	25.2	1.05 14	1.65 22	5.64 16	0.29 50	0.81 65	3.07 51	4.56 14	9.81 24	12.2 14	2.17 4	8.02 23	6.42 5	4.64
WarpMat [55]	27.6	1.16 22	1.35 7	6.04 28	0.18 21	0.24 7	2.44 34	5.02 21	9.30 21	13.0 23	3.49 54	8.47 34	9.01 59	4.98
ObjectStereo [98]	28.7	1.22 28	1.62 18	6.36 34	0.59 78	0.69 57	4.61 79	4.13 11	7.59 10	11.2 12	2.20 5	6.99 8	6.36 4	4.46
PMF [138]	31.3	1.74 65	2.04 54	8.07 74	0.33 56	0.49 38	4.16 72	**2.52** 1	5.87 3	**8.30** 1	2.13 3	6.80 6	6.32 3	4.06
LAMC-DSM [145]	31.3	1.61 59	2.18 61	5.86 25	0.24 38	0.60 46	3.12 52	4.63 15	10.4 29	12.7 19	2.09 2	8.31 28	6.10 2	4.83
PatchMatch [112]	32.6	2.09 84	2.33 67	9.31 83	0.21 27	0.39 23	2.62 43	2.99 3	8.16 11	9.62 5	2.47 14	7.80 15	7.11 16	4.59
HEBF [123]	32.7	1.10 20	1.38 9	5.74 21	0.22 31	0.33 18	2.41 32	6.54 54	11.8 49	15.2 38	2.78 26	9.28 57	8.10 37	5.41
HistoAggr2 [144]	32.7	1.93 74	2.30 66	6.39 35	0.16 16	0.46 35	2.22 25	5.88 32	11.3 38	14.7 32	2.41 12	7.78 14	6.89 13	5.20
imprNLCA [143]	33.2	1.38 42	1.83 40	7.38 58	0.21 29	0.41 26	2.26 27	5.99 34	11.5 42	14.3 29	2.85 32	6.68 5	7.98 34	5.23
GC+SegmBorder [57]	33.6	1.47 52	1.82 36	7.86 68	0.19 23	0.31 14	2.44 34	4.25 13	5.55 2	10.9 10	4.99 97	5.78 2	8.66 52	4.52
PMBP [131]	34.2	1.96 76	2.21 63	9.22 81	0.30 51	0.49 36	3.57 66	2.88 2	8.57 16	8.99 3	2.22 6	6.64 4	6.48 6	4.46
CrossLMF [126]	34.4	2.46 92	2.78 80	6.26 32	0.27 46	0.38 22	2.15 21	5.50 27	10.6 31	14.2 26	2.34 8	7.82 17	6.80 11	5.13
HighOrderBP [140]	34.7	1.56 57	2.12 60	7.99 73	0.14 12	0.26 10	1.65 10	4.97 18	10.6 30	12.3 16	3.22 49	8.48 35	8.34 46	5.14
Undr+OvrSeg [48]	35.6	1.89 72	2.22 65	7.22 55	0.11 6	0.22 6	1.34 5	6.51 50	9.98 26	16.4 54	2.92 37	8.00 22	7.90 29	5.39
DTAggr-P [141]	36.0	1.75 67	2.10 57	7.09 52	0.24 40	0.45 31	2.59 39	5.70 30	11.5 41	13.9 25	2.49 16	7.82 16	7.30 18	5.24
CostFilter [95]	38.0	1.51 56	1.85 43	7.61 63	0.20 26	0.39 24	2.42 33	6.16 40	11.8 51	16.0 47	2.71 23	8.24 26	7.66 24	5.55
SRW [139]	38.0	2.02 81	2.77 79	8.58 77	0.21 28	0.68 53	2.31 29	3.87 8	9.47 22	9.34 4	2.67 22	8.28 27	7.74 26	4.83
FeatureGC [107]	38.3	1.08 18	1.59 17	5.82 24	**0.08** 1	0.16 2	1.11 2	7.17 69	8.25 12	18.5 85	4.33 84	9.40 63	11.1 83	5.72
InfoPermeable [109]	39.1	1.06 16	1.53 15	5.64 16	0.32 53	0.88 71	4.15 71	5.60 28	13.0 68	14.5 30	2.65 21	9.16 55	7.69 25	5.51
AdaptOvrSegBP [33]	39.8	1.69 61	2.04 55	5.64 16	0.14 11	0.20 3	1.47 7	7.04 66	11.1 36	16.4 56	3.60 60	8.96 52	8.84 55	5.59

GlobalGCP [104]	40.8	0.87 3	2.54 74	4.69 5	0.16 18	0.53 41	2.22 25	6.44 47	11.5 44	16.2 50	3.59 58	9.49 66	8.95 58	5.60
NonLocalFilter [130]	40.9	1.47 52	1.85 43	7.88 70	0.25 43	0.42 27	2.60 41	6.01 36	11.6 47	14.3 28	2.87 33	8.45 33	8.10 38	5.48
CSM [120]	41.9	**0.82** 1	1.20 2	4.39 2	0.34 57	0.61 47	2.55 38	7.67 78	12.4 65	17.2 69	3.33 51	9.35 61	7.96 32	5.65
HistAggrSlant [121]	41.9	2.25 87	2.50 72	9.77 88	0.29 49	0.37 20	3.30 58	3.44 5	8.82 17	9.77 7	2.90 36	8.40 31	7.97 33	4.98
GeoSup [64]	42.6	1.45 48	1.83 41	7.71 65	0.14 14	0.26 9	1.90 17	6.88 62	13.2 72	16.1 49	2.94 39	8.89 50	8.32 45	5.80
SymBP+occ [7]	42.7	0.97 10	1.75 32	5.09 12	0.16 15	0.33 17	2.19 24	6.47 49	10.7 33	17.0 65	4.79 92	10.7 83	10.9 80	5.92
PlaneFitBP [32]	43.2	0.97 12	1.83 38	5.26 13	0.17 19	0.51 40	1.71 11	6.65 55	12.1 60	14.7 33	4.17 81	10.7 81	10.6 75	5.78
P-LinearS [99]	43.3	1.10 19	1.67 24	5.92 26	0.53 74	0.89 73	5.71 84	6.69 57	12.0 58	15.9 46	2.60 18	8.44 32	6.71 9	5.68
GAOH [136]	44.0	1.26 31	1.76 33	**4.31** 1	0.20 24	0.42 28	2.03 20	7.52 74	12.3 63	18.1 80	3.94 74	8.59 38	9.32 62	5.81
BSM [124]	45.4	3.08 107	3.38 88	7.80 66	0.26 45	0.70 58	2.40 30	5.74 31	8.95 19	14.8 36	2.34 9	8.79 46	6.80 10	5.42
AdaptLocalSeg [125]	46.4	1.33 40	1.82 36	7.19 54	0.32 54	0.79 62	4.50 75	5.32 25	11.9 56	14.5 31	2.73 24	9.69 69	7.91 31	5.67
ASSM [97]	47.3	1.28 36	1.69 27	6.44 38	0.64 81	1.11 82	5.61 83	6.51 50	11.5 43	16.4 57	2.81 30	8.07 24	7.19 17	5.77
AdaptDispCalib [36]	47.8	1.19 25	1.42 11	6.15 30	0.23 34	0.34 19	2.50 37	7.80 79	13.6 78	17.3 73	3.62 61	9.33 59	9.72 67	6.10
ConfSuppWin [113]	47.8	1.28 36	1.83 38	6.65 40	0.28 47	0.65 52	3.29 57	6.88 62	11.4 40	15.4 40	3.64 62	8.60 39	9.09 60	5.75
GeoDif [103]	47.8	1.88 71	2.35 68	7.64 64	0.38 63	0.82 66	3.02 50	5.99 35	11.3 37	13.3 24	2.84 31	8.33 29	8.09 36	5.49
RandomVote [89]	48.2	4.85 129	5.54 121	17.7 126	0.13 10	0.45 32	1.86 14	5.40 26	9.54 23	14.8 35	2.62 20	7.93 20	7.54 23	6.53
Segm+visib [4]	48.4	1.30 39	1.57 16	6.92 50	0.79 87	1.06 80	6.76 92	5.00 19	6.54 7	12.3 15	3.72 64	8.62 41	10.2 71	5.40
C-SemiGlob [19]	50.1	2.61 96	3.29 87	9.89 90	0.25 42	0.57 43	3.24 54	5.14 23	11.8 50	13.0 21	2.77 25	8.35 30	8.20 40	5.76
SCoBeP [132]	51.2	1.47 51	2.01 52	7.92 72	0.24 36	0.62 49	3.28 55	6.22 43	11.7 48	15.7 44	3.49 54	8.84 48	9.32 63	5.90
IterAdaptWgt [105]	51.3	0.85 2	1.28 3	4.59 3	0.35 60	0.86 68	4.53 77	7.60 75	14.5 95	17.3 75	3.20 48	9.36 62	8.49 48	6.08
ConvexOpt [118]	51.4	1.50 55	1.68 26	7.81 67	0.24 35	0.31 15	2.15 21	6.18 42	10.1 27	15.2 37	5.36 105	10.8 84	12.9 103	6.19
SO+borders [29]	51.5	1.29 38	1.71 28	6.83 46	0.25 44	0.53 42	2.26 27	7.02 65	12.2 61	16.3 52	3.90 70	9.85 73	10.2 72	6.03
RecursiveBF [122]	51.7	1.85 70	2.51 73	7.45 60	0.35 59	0.88 72	3.01 48	6.28 45	12.1 59	14.3 27	2.80 29	8.91 51	7.79 27	5.68
MultiResGC [49]	52.1	0.90 7	1.32 6	4.82 8	0.45 70	0.84 67	3.32 59	6.46 48	11.8 52	17.0 66	4.34 85	10.5 80	10.7 77	6.04
MVSegBP [66]	52.5	1.06 15	2.78 80	5.57 14	0.20 25	0.61 48	2.02 19	6.53 52	11.3 39	14.8 34	5.29 102	11.3 89	14.5 113	6.34
MSWLinRegr [128]	53.8	1.46 50	1.72 29	7.89 71	0.57 77	0.92 75	6.71 90	6.11 39	11.0 34	15.6 42	3.12 46	8.76 44	8.52 49	6.04
iFBS [115]	54.1	1.78 69	2.10 56	7.57 61	0.31 52	0.50 39	2.17 23	7.94 83	12.8 66	17.1 67	3.07 43	8.73 43	8.46 47	6.05
DistinctSM [27]	55.3	1.21 27	1.75 31	6.39 35	0.35 61	0.69 56	2.63 44	7.45 72	13.0 69	18.1 79	3.91 71	9.91 75	8.32 44	6.14
LocallyConsist [69]	55.3	1.70 63	2.21 62	5.67 19	0.16 17	0.32 16	1.63 9	8.68 98	13.9 83	17.0 64	4.19 82	10.8 85	9.72 66	6.33
CurveletSupWgt [73]	55.9	1.40 46	1.84 42	7.42 59	1.00 94	1.11 84	4.42 74	7.85 80	8.84 18	16.8 63	3.82 67	6.22 3	8.24 41	5.75
SegmentSupport [28]	57.7	1.25 30	1.62 20	6.68 41	0.25 41	0.64 50	2.59 39	8.43 93	14.2 90	18.2 81	3.77 65	9.87 74	9.77 68	6.44
RegionTreeDP [18]	58.0	1.39 45	1.64 21	6.85 48	0.22 33	0.57 43	1.93 18	7.42 71	11.9 55	16.8 61	6.31 115	11.9 96	11.8 90	6.56
OverSegmBP [26]	58.8	1.69 62	1.97 49	8.47 76	0.51 73	0.68 54	4.69 80	6.74 59	11.9 57	15.8 45	3.19 47	8.81 47	8.89 56	6.11
SNCC+AM [119]	59.8	3.21 109	3.57 93	13.6 108	0.22 32	0.45 30	3.01 48	6.41 46	10.4 28	17.7 77	3.11 45	8.61 40	9.27 61	6.63
RTAdaptWgt [114]	60.0	1.45 49	1.99 50	7.59 62	0.40 65	0.81 64	3.38 61	7.65 77	13.3 75	16.2 51	3.48 53	9.34 60	8.81 53	6.20
CostAggr+occ	60.4	1.38 43	1.96	7.14	0.44 69	1.13	4.87	6.80 60	11.9	17.3	3.60 59	8.57	9.36	6.20

[39]			48	53		85	81		54	72		37	64	
VSW [108]	62.2	1.62 60	1.88 47	6.98 51	0.47 72	0.81 63	3.40 62	8.67 97	13.3 77	18.0 78	3.37 52	8.85 49	8.12 39	6.29
EnhancedBP [24]	64.2	0.94 8	1.74 30	5.05 11	0.35 62	0.86 69	4.34 73	8.11 89	13.3 74	18.5 86	5.09 100	11.1 88	11.0 81	6.69
RealtimeHD [134]	65.1	2.16 85	2.46 70	10.1 92	0.24 37	0.44 29	3.40 62	6.27 44	10.7 32	16.6 60	4.70 91	10.1 78	12.8 101	6.66
PUTv3 [63]	65.8	1.77 68	3.86 98	9.42 85	0.42 67	0.95 77	5.72 85	7.02 64	14.2 89	18.3 83	2.40 11	9.11 54	6.56 8	6.64
GradAdaptWgt [60]	67.0	2.26 89	2.63 75	8.99 80	0.99 92	1.39 90	4.92 82	8.00 85	13.1 71	18.6 88	2.61 19	7.67 12	7.43 21	6.55
MultiCue [51]	67.5	1.20 26	1.81 35	6.31 33	0.43 68	0.69 55	3.36 60	7.09 67	14.0 87	17.2 71	5.42 108	12.6 100	12.5 100	6.89
SegTreeDP [22]	68.3	2.21 86	2.76 78	10.3 93	0.46 71	0.60 45	2.44 34	9.58 105	15.2 102	18.4 84	3.23 50	7.86 18	8.83 54	6.82
RT-ColorAW [106]	68.8	1.40 46	3.08 85	5.81 23	0.72 84	1.71 95	3.80 69	6.69 56	14.0 86	15.3 39	4.03 77	11.9 95	10.2 70	6.55
AdaptWeight [12]	68.9	1.38 43	1.85 45	6.90 49	0.71 83	1.19 87	6.13 86	7.88 81	13.3 76	18.6 89	3.97 75	9.79 71	8.26 42	6.67
InteriorPtLP [34]	69.7	1.27 33	1.62 18	6.82 45	1.15 100	1.67 94	12.7 109	8.07 87	11.9 53	18.7 91	3.92 73	9.68 68	9.62 65	7.26
ImproveSubPix [25]	73.3	3.00 105	3.61 94	10.9 99	0.88 89	1.47 91	7.10 95	7.12 68	12.4 64	16.6 59	2.96 41	8.22 25	8.55 50	6.90
RealTimeABW [81]	74.9	1.26 32	1.67 25	6.83 46	0.33 55	0.65 51	3.56 65	10.7 116	18.3 120	23.3 116	4.81 93	12.6 102	10.7 78	7.90
BP+DirectedDiff [61]	75.0	2.90 102	4.47 107	15.1 119	0.65 82	1.20 88	4.52 76	5.07 22	14.7 96	15.7 43	2.94 40	12.6 103	7.50 22	7.29
SemiGlob [6]	76.8	3.26 110	3.96 99	12.8 105	1.00 93	1.57 92	11.3 102	6.02 37	12.2 62	16.3 53	3.06 42	9.75 70	8.90 57	7.50
SDDS [133]	78.2	3.31 111	3.62 95	10.4 95	0.39 64	0.76 60	2.85 46	7.65 76	13.0 67	19.4 93	3.99 76	10.00 77	10.8 79	7.19
FastBilateral [68]	78.7	2.38 91	2.80 82	10.4 94	0.34 58	0.92 74	4.55 78	9.83 110	15.3 103	20.3 101	3.10 44	9.31 58	8.59 51	7.31
HistoAggr [111]	81.4	2.47 93	2.71 77	11.1 100	0.74 85	0.97 79	3.28 55	8.31 91	13.8 81	21.0 109	3.86 68	9.47 65	10.4 74	7.33
PlaneFitSGM [83]	82.1	3.13 108	4.20 102	14.9 116	1.08 97	1.87 98	14.6 115	5.68 29	11.6 46	17.1 68	3.79 66	9.26 56	11.3 84	8.21
RealtimeBFV [65]	82.5	1.71 64	2.22 64	6.74 42	0.55 76	0.87 70	2.88 47	9.90 111	15.0 99	19.5 94	6.66 119	12.3 98	13.4 106	7.65
2OP+occ [37]	82.6	2.91 103	3.56 92	7.33 57	0.24 38	0.49 37	2.76 45	10.9 118	15.4 104	20.6 104	5.42 107	10.8 87	12.5 99	7.75
CostRelaxAW [59]	84.1	2.91 104	3.49 91	11.4 101	0.60 79	1.11 83	6.45 89	7.92 82	13.7 79	20.9 107	3.59 57	9.43 64	10.3 73	7.66
VariableCross [44]	84.7	1.99 78	2.65 76	6.77 43	0.62 80	0.96 78	3.20 53	9.75 106	15.1 100	18.2 82	6.28 113	12.7 105	12.9 102	7.60
BPcompressed [56]	84.8	2.68 97	3.63 96	9.59 86	1.33 104	1.89 100	9.09 100	8.36 92	13.9 84	16.4 55	3.71 63	9.85 72	9.92 69	7.53
RealtimeBP [21]	84.9	1.49 54	3.40 90	7.87 69	0.77 86	1.90 101	9.00 99	8.72 99	13.2 73	17.2 70	4.61 88	11.6 92	12.4 98	7.69
CCH+SegAggr [47]	86.6	1.74 65	2.11 58	9.23 82	0.41 66	0.94 76	3.97 70	8.08 88	14.3 91	19.8 97	7.07 121	12.9 106	16.3 119	8.07
FastAggreg [45]	88.1	1.16 23	2.11 59	6.06 29	4.03 130	4.75 127	6.43 88	9.04 102	15.2 101	20.2 99	5.37 106	12.6 101	11.9 92	8.24
GC+occ [2]	88.2	1.19 24	2.01 53	6.24 31	1.64 113	2.19 106	6.75 91	11.2 122	17.4 115	19.8 96	5.36 104	12.4 99	13.0 104	8.26
MultiCamGC [3]	88.6	1.27 34	1.99 50	6.48 39	2.79 123	3.13 116	3.60 67	12.0 124	17.6 117	22.0 111	4.89 94	11.8 94	12.1 94	8.31
VarMSOH [54]	88.6	3.97 119	5.23 116	14.9 117	0.28 47	0.76 61	3.78 68	9.34 104	14.3 92	20.0 98	4.14 79	9.91 76	11.4 86	8.17
Unsupervised [74]	91.1	3.89 118	4.39 105	18.8 129	1.01 95	1.14 86	11.3 102	6.72 58	6.98 8	16.1 48	9.93 129	10.7 82	22.5 133	9.45
Layered [5]	91.3	1.57 58	1.87 46	8.28 75	1.34 105	1.85 97	6.85 93	8.64 96	14.3 93	18.5 87	6.59 118	14.7 117	14.4 111	8.24
SNCC [77]	92.5	5.17 134	6.08 127	21.7 136	0.95 91	1.73 96	12.0 106	8.04 86	11.1 35	22.9 114	3.59 56	9.02 53	10.7 76	9.41
ESAW [86]	93.1	1.92 73	2.45 69	9.66 87	1.03 96	1.65 93	6.89 94	8.48 94	14.2 88	18.7 90	6.56 117	12.7 104	14.4 112	8.21
OptimizedDP [70]	95.2	1.97 77	3.78 97	9.80 89	3.33 126	4.74 126	13.0 111	6.53 53	13.9 85	16.6 58	5.17 101	13.7 112	13.4 107	8.83
AdaptPolygon [43]	95.3	2.29 90	2.88 84	8.94 79	0.80 88	1.11 81	3.41 64	10.5 114	15.9 109	21.3 110	6.13 112	13.2 108	13.3 105	8.32

StereoSONN [71]	95.5	4.04 121	4.74 109	18.1 128	0.53 75	0.75 59	6.21 87	8.53 95	13.7 80	20.2 99	5.07 98	10.8 86	14.0 109	8.89
RealtimeVar [72]	95.8	3.33 112	5.48 119	16.8 125	1.15 101	2.35 107	12.8 110	6.18 41	13.1 70	17.3 74	4.66 90	11.7 93	13.7 108	9.05
ConvexTV [46]	96.8	3.61 115	5.72 122	18.0 127	1.16 102	2.50 110	12.4 108	6.10 38	15.7 106	16.8 62	3.88 69	14.4 115	11.5 87	9.30
GenModel [20]	100.2	2.57 95	4.74 110	13.0 106	1.72 114	3.08 114	16.9 119	6.86 61	15.0 98	19.2 92	4.64 89	14.9 119	11.4 85	9.50
TensorVoting [9]	102.1	3.79 117	4.79 111	8.86 78	1.23 103	1.88 99	11.5 104	9.76 107	17.0 113	24.0 121	4.38 86	11.4 90	12.2 96	9.25
RealTimeGPU [14]	102.4	2.05 83	4.22 103	10.6 98	1.92 117	2.98 113	20.3 127	7.23 70	14.4 94	17.6 76	6.41 116	13.7 111	16.5 121	9.82
RTCensus [50]	102.9	5.08 133	6.25 129	19.2 131	1.58 112	2.42 108	14.2 114	7.96 84	13.8 82	20.3 102	4.10 78	9.54 67	12.2 95	9.73
ReliabilityDP [13]	104.0	1.36 41	3.39 89	7.25 56	2.35 120	3.48 123	12.2 107	9.82 109	16.9 112	19.5 95	12.9 136	19.9 135	19.7 125	10.7
TwoWin [92]	104.0	2.25 87	3.08 86	11.6 102	0.92 90	1.31 89	7.53 96	10.7 117	15.8 108	23.6 118	8.25 124	13.5 109	16.6 122	9.59
LDE [142]	104.8	2.68 97	4.49 108	14.3 112	1.12 99	2.12 105	14.9 116	8.88 101	17.7 118	22.5 113	4.57 87	14.0 113	11.8 89	9.91
HRMBIL [93]	104.9	3.60 114	4.79 111	16.8 124	1.38 107	2.72 111	17.3 120	7.48 73	15.8 107	20.8 106	4.29 83	13.7 110	12.0 93	10.0
CostRelax [11]	107.7	4.76 127	6.08 128	20.3 134	1.41 109	2.48 109	18.5 124	8.18 90	15.9 110	23.8 119	3.91 72	10.2 79	11.8 91	10.6
AdaptDomainBP [100]	110.1	4.82 128	7.10 133	16.1 121	1.57 111	2.82 112	13.6 112	9.13 103	17.4 115	20.9 108	4.16 80	14.4 116	11.0 82	10.2
DOUS-Refine [87]	110.8	2.85 99	4.28 104	13.2 107	2.30 119	3.17 118	15.6 117	8.76 100	15.0 97	20.3 103	8.02 123	14.8 118	19.5 124	10.6
TreeDP [8]	111.4	1.99 79	2.84 83	9.96 91	1.41 108	2.10 104	7.74 97	15.9 132	23.9 133	27.1 128	10.0 130	18.3 129	18.9 123	11.7
GC [1d]	112.8	1.94 75	4.12 100	9.39 84	1.79 116	3.44 122	8.75 98	16.5 133	25.0 137	24.9 122	7.70 122	18.2 128	15.3 116	11.4
CSBP [82]	113.0	2.00 80	4.17 101	10.5 97	1.48 110	3.11 115	17.7 122	11.1 121	20.2 126	27.5 130	5.98 111	16.5 125	16.0 118	11.4
DCBGrid [88]	114.3	5.90 138	7.26 137	21.0 135	1.35 106	1.91 102	11.2 101	10.5 113	17.2 114	22.2 112	5.34 103	11.9 97	14.9 114	10.9
BioPsyASW [80]	114.5	3.62 116	5.52 120	14.6 114	3.15 124	4.20 125	20.4 128	11.5 123	18.2 119	23.2 115	4.93 95	13.0 107	11.7 88	11.2
HBpStereoGpu [101]	114.9	3.37 113	5.34 118	13.6 109	1.12 98	2.06 103	14.1 113	12.2 125	19.0 123	27.2 129	6.29 114	14.2 114	16.4 120	11.2
BP+MLH [40]	115.9	4.17 123	6.34 130	14.6 115	1.96 118	3.31 120	16.8 118	10.2 112	18.9 122	24.0 120	4.93 96	15.5 120	12.3 97	11.1
H-Cut [76]	116.3	2.85 100	4.86 114	14.4 113	1.73 115	3.14 117	20.2 126	10.7 115	19.5 124	25.8 125	5.46 109	15.6 121	15.7 117	11.7
SAD-IGMCT [52]	119.1	5.81 137	7.14 135	22.6 138	2.61 122	3.33 121	25.3 134	9.79 108	15.5 105	25.7 124	5.08 99	11.5 91	15.0 115	12.5
FLTG-DDE [90]	120.4	3.03 106	5.28 117	15.0 118	3.39 127	5.02 129	25.0 133	11.0 119	19.5 125	26.3 127	5.78 110	16.0 124	14.2 110	12.5
DPVI [67]	125.0	4.76 126	5.83 123	16.6 123	4.89 132	5.66 132	22.9 131	11.0 120	16.2 111	23.4 117	9.64 127	15.6 122	23.5 136	13.3
DP [1b]	125.2	4.12 122	5.04 115	12.0 103	10.1 141	11.0 141	21.0 129	14.0 127	21.6 128	20.6 104	10.5 132	19.1 131	21.1 129	14.2
2DPOC [110]	125.2	2.88 101	4.80 113	10.5 96	6.55 133	7.82 134	17.4 121	14.4 128	22.1 129	27.9 131	15.2 141	22.7 139	24.5 137	14.7
Bipartite [78]	126.5	2.54 94	4.41 106	13.6 110	6.62 134	7.46 133	18.6 125	16.9 135	24.1 135	30.2 133	15.1 140	21.8 138	23.0 135	15.4
PhaseBased [31]	130.2	4.26 124	6.53 131	15.4 120	6.71 135	8.16 135	26.4 136	14.5 129	23.1 130	25.5 123	10.8 134	20.5 136	21.2 130	15.3
RegionalSup [38]	131.1	3.99 120	6.05 126	14.2 111	8.14 137	9.68 138	36.8 142	18.3 140	26.7 140	32.1 134	9.16 126	19.3 132	19.9 127	17.0
IMCT [62]	131.3	4.54 125	5.90 125	19.8 133	3.16 125	3.83 124	23.2 132	18.0 138	23.1 131	35.3 138	12.7 135	18.5 130	27.9 140	16.3
BioDEM [117]	131.4	6.57 141	8.43 140	28.1 142	3.61 128	4.80 128	33.7 139	13.2 126	21.3 127	34.5 137	6.84 120	16.0 123	19.8 126	16.4
SSD+MF [1a]	131.8	5.23 135	7.07 132	24.1 140	3.74 129	5.16 130	11.9 105	16.5 134	24.8 136	32.9 136	10.6 133	19.8 133	26.3 138	15.7
FW-DLR [129]	132.0	4.87 130	5.89 124	22.9 139	2.50 121	3.22 119	18.3 123	18.2 139	18.7 121	37.2 140	24.2 143	27.9 142	42.1 143	18.8
SO [1c]	133.4	5.08 132	7.22 136	12.2 104	9.44 140	10.9 140	21.9 130	19.9 141	28.2 143	26.3 126	13.0 137	22.8 140	22.3 132	16.6
MI-nonpara [85]	135.8	5.59 136	7.54 138	18.8 130	7.50 136	8.99 136	35.0 140	17.4 136	25.7 139	36.9 139	10.2 131	19.9 134	22.6 134	18.0
PhaseDiff [23]	136.7	4.89 131	7.11 134	16.3 122	8.34 139	9.76 139	26.0 135	20.0 142	28.0 142	29.0 132	19.8 142	28.5 143	27.5 139	18.8

Rank+ASW [84]	136.9	6.51 (140)	8.43 (140)	19.7 (132)	10.5 (142)	12.0 (142)	32.7 (138)	15.7 (130)	24.1 (134)	32.8 (135)	14.1 (138)	23.1 (141)	21.7 (131)	18.4
STICA [16]	137.0	7.70 (142)	9.63 (143)	27.8 (141)	8.19 (138)	9.58 (137)	40.3 (143)	15.8 (131)	23.2 (132)	37.7 (141)	9.80 (128)	17.8 (127)	28.7 (141)	19.7
LCDM+AdaptWgt [75]	137.2	5.98 (139)	7.84 (139)	22.2 (137)	14.5 (143)	15.4 (143)	35.9 (141)	20.8 (143)	27.3 (141)	38.3 (142)	8.90 (125)	17.2 (126)	20.0 (128)	19.5
Infection [10]	138.6	7.95 (143)	9.54 (142)	28.9 (143)	4.41 (131)	5.53 (131)	31.7 (137)	17.7 (137)	25.1 (138)	44.4 (143)	14.3 (139)	21.3 (137)	38.0 (142)	20.7

Abbildung A.-4.: Middlebury Ranking Stand: Februar 2013

Literatur

[1] E. Ahmed und J. Rose. „The effect of LUT and cluster size on deep-submicron FPGA performance and density“. In: *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*. Monterey, California, United States: ACM, 2000, S. 3–12. DOI: http://doi.acm.org/10.1145/329166.329171 (siehe S. 54).

[2] C. Albrecht. *IWLS 2005 Benchmarks*. 2005. URL: http://www.iwls.org/iwls2005/benchmarks.html (siehe S. 58).

[3] Altera. *CPLDs vs. FPGAs: Comparing High-Capacity Programmable Logic*. Techn. Ber. 18. San Jose, 1995, S. 12 (siehe S. 53, 55).

[4] Altera. *Guidance for Accurately Benchmarking FPGAs*. San Jose, 2007 (siehe S. 58).

[5] K. Ambrosch, M. Humenberger, W. Kubinger und A. Steininger. „SAD-Based Stereo Matching Using FPGAs“. In: *Embedded Computer Vision*. Springer London, 2009, S. 121–138. URL: http://www.springerlink.com/content/k541v3j012028m43/ (siehe S. 27, 71).

[6] K. Asanovic, R. Bodik, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams und K. Yelick. *The landscape of parallel computing research: a view from Berkeley*. Techn. Ber. 2006. DOI: citeulike-article-id:1671417. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf (siehe S. 58).

[7] C. Banz, S. Hesselbarth, H. Flatt, H. Blume und P. Pirsch. „Hardware-Architektur zur echtzeitfähigen Berechnung dichter Disparitätskarten“. In: *International Conference on embedded Computer Systems: Architectures, Modeling and Simulation*. Greece: IEEE, 2010 (siehe S. 73, 128).

[8] S. Birchfield und C. Tomasi. „Depth discontinuities by pixel-to-pixel stereo“. In: *International Journal of Computer Vision* 35.3 (1999), S. 269–293. URL: %3CGo%20to%20ISI%3E://000084252400004 (siehe S. 37).

[9] A. F. Bobick und S. S. Intille. „Large Occlusion Stereo“. In: *Int. J. Comput. Vision* 33.3 (1999), S. 181–200. DOI: http://dx.doi.org/10.1023/A:1008150329890 (siehe S. 37).

[10] J. Bodily, B. Nelson, Z. Wei, D.-J. Lee und J. Chase. „A Comparison Study on Implementing Optical Flow and Digital Communications on FPGAs and GPUs“. In: *ACM Trans. Reconfigurable Technol. Syst.* 3.2 (2010), S. 1–22. DOI: http://doi.acm.org/10.1145/1754386.1754387 (siehe S. 58).

[11] A. Börner, H. Hirschmüller, K. Scheibe, M. Suppa und J. Wohlfeil. „MFC - A Modular Line Camera for 3D World Modulling“. In: *Robot Vision*. Hrsg.

von G. Sommer und R. Klette. Bd. 4931. Springer Berlin / Heidelberg, 2008, S. 319–326. DOI: 10.1007/978-3-540-78157-8. URL: http://dx.doi.org/10.1007/978-3-540-78157-8_24 (siehe S. 44).

[12] Y. Boykov, O. Veksler und R. Zabih. „Fast Approximate Energy Minimization via Graph Cuts“. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.11 (Nov. 2001), S. 1222–1239. ISSN: 0162-8828. DOI: 10.1109/34.969114. URL: http://dx.doi.org/10.1109/34.969114 (siehe S. 30).

[13] M. Buder. „Dense real-time stereo matching using memory efficient semi-global-matching variant based on FPGAs“. In: Hrsg. von K. Nasser und F. C. Matthias. Bd. 8437. 1. SPIE, 2012, S. 843709. URL: http://link.aip.org/link/?PSI/8437/843709/1%20http://dx.doi.org/10.1117/12.921147 (siehe S. 105).

[14] S. Che, J. Li, J. W. Sheaffer, K. Skadron und J. Lach. „Accelerating Compute-Intensive Applications with GPUs and FPGAs“. In: *Proceedings of the 2008 Symposium on Application Specific Processors*. IEEE Computer Society, 2008, S. 101–107. DOI: http://dx.doi.org/10.1109/SASP.2008.4570793 (siehe S. 59).

[15] S.-Y. Cho und R. K. Gupta. *Real-time Stereo Matching using Adaptive BinaryWindow*. 2010 (siehe S. 28, 70, 121).

[16] L. Chunho, M. Potkonjak und W. H. Mangione-Smith. „MediaBench: a tool for evaluating and synthesizing multimedia and communications systems“. In: *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*. 1997, S. 330–335. URL: 10.1109/MICRO.1997.645830 (siehe S. 58).

[17] M. College. *Middlebury Stereo Evaluation - Version 2*. Hrsg. von M. S. Vision. Middlebury UK, 2010. URL: http://vision.middlebury.edu/stereo/ (siehe S. 90, 91, 97).

[18] P. d'Angelo und P. Reinartz. „Semiglobal Matching Results on the ISPRS Stereo Matching Benchmark“. In: *High-Resolution Earth Imaging for Geospatial Information*. Hrsg. von C. Heipke, K. Jacobsen, F. Rottensteiner, S. Müller und U. Sörgel. Hannover, Germany: Institute of Photogrammetry und GeoInformation, Leibniz Universität Hannover, 2011. URL: http://elib.dlr.de/70796/ (siehe S. 73).

[19] V. Design. *Videre Desing*. 2010. URL: http://www.videredesign.com/ (siehe S. 74, 76).

[20] EMBC. *The Embedded Microprocessor Benchmark Consortium*. 2010. URL: http://www.eembc.org (siehe S. 58).

[21] R. Enzler, T. Jeger, D. Cottet und G. Tröster. „High-Level Area and Performance Estimation of Hardware Building Blocks on FPGAs“. In: *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing*. Hrsg. von R. Hartenstein und H. Grünbacher. Bd. 1896. Springer Berlin / Heidelberg, 2000, S. 525–534. DOI: 10.1007/3-540-44614-1. URL: http://dx.doi.org/10.1007/3-540-44614-1_57 (siehe S. 58).

[22] I. Ernst und H. Hirschmüller. „Mutual Information Based Semi-Global Stereo Matching on the GPU“. In: *ISVC '08: Proceedings of the 4th International Symposium on Advances in Visual Computing*. Berlin, Heidelberg: Springer-Verlag, 2008, S. 228–239 (siehe S. 73, 74, 111).

[23] S. A. Fahmy. „Novel FPGA-based implementation of median and weighted median filters for image processing“. In: Hrsg. von P. Y. K. Cheung und W. Luk. 2005, S. 142–147. URL: http://doi.ieeecomputersociety.org/10.1109/FPL.2005.1515713 (siehe S. 92).

[24] O. Faugeras, T. Vie ville, E. Theron, J. Vuillemin, B. Hotz, Z. Zhang, L. Moll, P. Bertin, H. Mathieu, P. Fua, G. Berry und C. Proy. *Real-time correlation-based stereo : algorithm, implementations and applications*. Techn. Ber. 1993. URL: http://hal.inria.fr/inria-00074658/PDF/RR-2013.pdf (siehe S. 71, 72).

[25] P. F. Felzenszwalb und D. P. Huttenlocher. „Efficient Belief Propagation for Early Vision“. In: *Int. J. Comput. Vision* 70.1 (2006), S. 41–54 (siehe S. 30, 37, 69).

[26] M. J. Flynn. „Very high-speed computing systems“. In: *Proceedings of the IEEE* 54.12 (1966), S. 1901–1909. URL: 10.1109/PROC.1966.5273 (siehe S. 49).

[27] S. Forstmann, Y. Kanou, J. Ohya, S. Thuering und A. Schmitt. „Real-Time Stereo by using Dynamic Programming“. In: *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3*. Washington, DC, USA: IEEE Computer Society, 2004, S. 29 (siehe S. 37, 70).

[28] A. Fusiello. „Efficient Stereo with Multiple Windowing“. In: Hrsg. von R. Vito und T. Emanuele. 1997, S. 858–858. URL: http://doi.ieeecomputersociety.org/10.1109/CVPR.1997.609428 (siehe S. 28).

[29] A. Fusiello, E. Trucco und A. Verri. „A compact algorithm for rectification of stereo pairs“. In: *Machine Vision and Applications* 12.1 (2000), S. 16–22. DOI: 10.1007/s001380050120. URL: http://dx.doi.org/10.1007/s001380050120 (siehe S. 20, 89).

[30] D. D. Gajski. *New VLSI Tools*. 1983. URL: http://doi.ieeecomputersociety.org/10.1109/MC.1983.1654264 (siehe S. 61).

[31] S. K. Gehrig, F. Eberli und T. Meyer. „A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching“. In: *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*. Liege, Belgium: Springer-Verlag, 2009, S. 134–143 (siehe S. 39, 73, 74, 121).

[32] S. K. Gehrig und C. Rabe. *Real-Time Semi-Global Matching on the CPU*. Techn. Ber. Sindelfingen, 2010 (siehe S. 73, 74).

[33] S. Gehrke, K. Morin, M. Downey, N. Boehrer und T. Fuchs. „Convergence in Geomatics – Shaping Canada's Competitive Landscape“. In: *The 2010 Canadian Geomatics Conference and Symposium of Commission I, ISPRS*. Hrsg. von ISPRS. Bd. ISPRS WG I.2. Calgary, Alberta, Canada: ISPRS, 2010 (siehe S. 73, 101).

[34] S. Geman und D. Geman. „Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images“. In: *Readings in uncertain reasoning*. Morgan Kaufmann Publishers Inc., 1990, S. 452–472 (siehe S. 29).

[35] C. Glennie und D. D. Lichti. „Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning“. In: *Remote Sensing* 2.6 (2010), S. 1610–1624. URL: http://www.mdpi.com/2072-4292/2/6/1610/ (siehe S. 85).

[36] Z. GmbH. *webpage*. 2011. URL: http://www.zf-laser.com/ (siehe S. 85).

[37] M. Gong, R. Yang, L. Wang und M. Gong. „A Performance Study on Different Cost Aggregation Approaches Used in Real-Time Stereo Matching“. In: *International Journal of Computer Vision* 75.2 (2007), S. 283–296. URL: http://www.springerlink.com/content/q723r01433637x77/ (siehe S. 29, 71).

[38] P. Graham und B. Nelson. „Genetic algorithms in software and in hardware-a performance analysis of workstation and custom computing machine implementations“. In: *FPGAs for Custom Computing Machines, 1996. Proceedings. IEEE Symposium on*. 1996, S. 216–225. URL: 10.1109/FPGA.1996.564847 (siehe S. 58).

[39] D. Grießbach, D. Baumbach, A. Börner, M. Buder, I. Ernst, E. Funk, J. Wohlfeil und S. Zuev. „IPS - A SYSTEM FOR REAL-TIME NAVIGATION AND 3D MODELING“. In: *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XXXIX-B5 (2012), S. 21–26. DOI: 10.5194/isprsarchives-XXXIX-B5-21-2012. URL: http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXIX-B5/21/2012/ (siehe S. 123).

[40] D. Griessbach, M. Bauer, M. Scheele, A. Hermerschmidty und S. Krueger. „Stereo camera calibration with diffractive optical elements“. In: *Stereo camera calibration with diffractive optical elements*. Deutsche Gesellschaft für angewandte Optik e. V. und Società Italiana di Ottica e Fotonica, 2009 (siehe S. 88, 89).

[41] D. Han und D.-H. Hwang. „A Novel Stereo Matching Method for Wide Disparity Range Detection“. In: *Image Analysis and Recognition*. Bd. 3656. Springer Berlin / Heidelberg, 2005, S. 643–650. URL: http://www.springerlink.com/content/dbnpq3cbkbug8gv6/ (siehe S. 71).

[42] S.-K. Han, M.-H. Jeong, S. Woo und B.-J. You. „Architecture and Implementation of Real-Time Stereo Vision with Bilateral Background Subtraction“. In: *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*. Bd. 4681. Springer Berlin / Heidelberg, 2007, S. 906–912. URL: http://www.springerlink.com/content/p45122x06lx6055x/ (siehe S. 71).

[43] R. Hartley und A. Zisserman. *Multiple view geometry in computer vision / Richard Hartley, Andrew Zisserman*. Hrsg. von A. Zisserman. Accessed from http://nla.gov.au/nla.cat-vn1112671. Cambridge : Cambridge University Press, 2003 (siehe S. 14, 20).

[44] E. Hecht. *Optics (4th Edition)*. Addison Wesley, 2001. DOI: citeulike-article-id:319275 (siehe S. 8).

[45] M. Heinrichs, V. Rodehorst und O. Hellwich. „EFFICIENT SEMI-GLOBAL MATCHING FOR TRINOCULAR STEREO“. In: *International Archives of Photogrammetry,* Remote Sensing and Spatial Information Sciences (2007) (siehe S. 73).

[46] S. Hermann und R. Klette. *The Naked Truth about Cost Functions.* 2009. URL: http://www.mi.auckland.ac.nz/tech-reports/MItech-TR-33.pdf (siehe S. 27).

[47] S. Hermann, R. Klette und E. Destefanis. „Inclusion of a Second-Order Prior into Semi-Global Matching“. In: *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology.* Tokyo, Japan: Springer-Verlag, 2008, S. 633–644. DOI: http://dx.doi.org/10.1007/978-3-540-92957-4_55 (siehe S. 73).

[48] H. Hirschmüller. „Stereo Vision in Structured Environments by Consistent Semi-Global Matching“. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on.* Bd. 2. 2006, S. 2386–2393. URL: 10.1109/CVPR.2006.294 (siehe S. 37, 73, 90, 121).

[49] H. Hirschmüller und S. Gehrig. „Stereo matching in the presence of sub-pixel calibration errors“. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* 2009, S. 437–444 (siehe S. 90).

[50] H. Hirschmüller. „Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information“. In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2.* Washington, DC, USA: IEEE Computer Society, 2005, S. 807–814 (siehe S. 37, 73, 90, 116).

[51] H. Hirschmüller. „Stereo Processing by Semiglobal Matching and Mutual Information“. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.2 (2008), S. 328–341. DOI: http://dx.doi.org/10.1109/TPAMI.2007.1166 (siehe S. 37, 39, 43, 90, 116).

[52] H. Hirschmüller. „Stereo Vision Based Mapping and Immediate Virtual Walkthroughs“. Diss. Leicester UK: De Montfort University, 2003 (siehe S. 35, 117).

[53] H. Hirschmüller, M. Buder und I. Ernst. „Memory Efficient Semi-Global Matching“. In: *ISPRS Annals of photogrammetry remote sensing and the spatial information sciences* (2012) (siehe S. 105, 106, 113, 116, 142).

[54] H. Hirschmüller, P. R. Innocent und J. Garibaldi. *Real-Time Correlation-Based Stereo Vision with Reduced Border Errors.* 2002 (siehe S. 27).

[55] H. Hirschmüller und D. Scharstein. „Evaluation of Stereo Matching Costs on Images with Radiometric Differences“. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 31.9 (2009), S. 1582–1599 (siehe S. 27, 73, 91).

[56] H. Hirschmüller, F. Scholten und G. Hirzinger. „Stereo Vision Based Reconstruction of Huge Urban Areas from an Airborne Pushbroom Camera (HRSC)“. In: *Pattern Recognition.* Hrsg. von W. G. Kropatsch, R. Sablatnig und A. Hanbury. Bd. 3663. Springer Berlin / Heidelberg, 2005, S. 58–66. DOI:

10.1007/11550518. URL: http://dx.doi.org/10.1007/11550518_8 (siehe S. 84).

[57] M. Humenberger, T. Engelke und W. Kubinger. „A Census-Based Stereo Vision Algorithm Using Modified Semi-Global Matching and Plane Fitting to Improve Matching Quality“. In: (2010). URL: http://robots-at-home.acin.tuwien.ac.at/publications/conferences/PID1250463.pdf (siehe S. 73).

[58] M. Humenberger, C. Zinner, M. Weber, W. Kubinger und M. Vincze. „A fast stereo matching algorithm suitable for embedded real-time systems“. In: *Computer Vision and Image Understanding* (2010). URL: http://dx.doi.org/10.1016/j.cviu.2010.03.012 (siehe S. 70, 121).

[59] B. Jahne, P. Geissler und H. Haussecker. *Handbook of Computer Vision and Applications*. 1. Morgan Kaufmann Publishers Inc., 1999, S. 2592 (siehe S. 4).

[60] H. Jeong und S. Park. „Generalized Trellis Stereo Matching with Systolic Array“. In: *Parallel and Distributed Processing and Applications*. Bd. 3358. Springer Berlin / Heidelberg, 2005, S. 263–267. URL: http://www.springerlink.com/content/6pb5rljk0g07tl9y/ (siehe S. 70).

[61] T. Kanade, A. Yoshida, K. Oda, H. Kano und M. Tanaka. „A Stereo Machine for Video-rate Dense Depth Mapping and Its New Applications“. In: *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*. 1996, S. 196–202 (siehe S. 34).

[62] P. Kannan und D. Bhatia. „Estimating Pre-Placement FPGA Interconnection Requirements“. In: *Proceedings of the 17th International Conference on VLSI Design*. IEEE Computer Society, 2004, S. 869 (siehe S. 57).

[63] Z. Ke, L. Jiangbo, G. Lafruit, R. Lauwereins und L. Van Gool. „Real-time accurate stereo with bitwise fast voting on CUDA“. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. 2009, S. 794–800. DOI: 10.1109/iccvw.2009.5457623 (siehe S. 70, 120, 121).

[64] A. Klaus, M. Sormann und K. Karner. „Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure“. In: *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, S. 15–18 (siehe S. 25, 69).

[65] V. Kolmogorov und R. Zabih. „Multi-camera scene reconstruction via graph cuts“. In: *Computer Vision - Eccv 2002 Pt Iii* 2352 (2002), S. 82–96. URL: %3CGo%20to%20ISI%3E://000181393900006 (siehe S. 30, 37).

[66] S. Kosov, T. Thormhlen und H.-P. Seidel. „Accurate Real-Time Disparity Estimation with Variational Methods“. In: *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*. Las Vegas, Nevada: Springer-Verlag, 2009, S. 796–807 (siehe S. 70, 121).

[67] A. Kristian. „Hardware implementation of an SAD based stereo vision algorithm“. In: Hrsg. von K. Wilfried, H. Martin und S. Andreas. 2007, S. 1–6. URL: http://doi.ieeecomputersociety.org/10.1109/CVPR.2007.383417 (siehe S. 27, 72).

[68] D. Krutz. *Ein Betriebssystem für konfigurierbare Hardware*. Berlin, 2006. URL: http://edoc.hu-berlin.de/docviews/abstract.php?id=27784 (siehe S. 60, 62, 63, 65–67).

[69] M. Kuhn, S. Moser, O. Isler, F. K. Gürkaynak, A. Burg, N. Felber, H. Kaeslin und W. Fichtner. *Efficient ASIC Implementation of a Real-Time Depth Mapping Stereo Vision System*. 2003 (siehe S. 71, 72).

[70] I. Kuon und J. Rose. „Measuring the Gap Between FPGAs and ASICs“. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 26.2 (2007), S. 203–215. URL: 10.1109/TCAD.2006.884574 (siehe S. 56, 58).

[71] I. Kuon, R. Tessier und J. Rose. „FPGA Architecture: Survey and Challenges“. In: *Found. Trends Electron. Des. Autom.* 2.2 (2008), S. 135–253. DOI: citeulike-article-id:5973049. URL: http://dx.doi.org/10.1561/1000000005 (siehe S. 57, 58).

[72] M. Li und Y. Jia. „Stereo Vision System on Programmable Chip (SVSoC) for Small Robot Navigation“. In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. 2006, S. 1359–1365. URL: 10.1109/IROS.2006.281923 (siehe S. 70).

[73] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen und H. Chen. „Hardware-efficient belief propagation“. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* (2009), S. 80–87 (siehe S. 30, 70).

[74] T. Liebeskind. „FPGA-BASIERTE ECHTZEIT-KORREKTUR VON ABBILDUNGSFEHLERN OPTISCHER SYSTEME“. Diss. Fachhochschule Nordhausen: Fachhochschule Nordhausen, 2009 (siehe S. 94).

[75] E. Lindholm. *NVIDIA Tesla: A Unified Graphics and Computing Architecture*. 2008. URL: http://doi.ieeecomputersociety.org/10.1109/MM.2008.31 (siehe S. 51).

[76] C. Loop und Z. Zhengyou. „Computing rectifying homographies for stereo vision“. In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* Bd. 1. 1999, 131 Vol. 1 (siehe S. 20).

[77] T. Luhmann. *Nahbereichsphotogrammetrie: Grundlagen, Methoden und Anwendungen*. Wichmann Herbert, 2010. URL: http://books.google.com/books?id=qdyxPwAACAAJ (siehe S. 15–17, 134).

[78] T. Luhmann. „Erweiterte Verfahren zur geometrischen Kamerakalibration in der Nahbereichsphotogrammetrie.pdf“. Diss. München, 2009 (siehe S. 33).

[79] J. Mallon und P. F. Whelan. „Projective rectification from the fundamental matrix“. In: *Image and Vision Computing* 23.7 (2005), S. 643–650. DOI: 10.1016/j.imavis.2005.03.002. URL: http://www.sciencedirect.com/science/article/pii/S0262885605000399 (siehe S. 20).

[80] D. K. Masrani und W. J. MacLean. „A Real-Time Large Disparity Range Stereo-System using FPGAs“. In: *ICVS '06: Proceedings of the Fourth IEEE International Conference on Computer Vision Systems*. Washington, DC, USA: IEEE Computer Society, 2006, S. 13 (siehe S. 72).

[81] V. Matrose. „Optimierung der Verdrahtbarkeit unter Berücksichtigung heterogener Verdrahtungsressourcen hierarchischer FPGA-Architekturen“. Diss. Berlin: Technischen Universität Berlin, 2009 (siehe S. 56).

[82] S. Mattoccia, F. Tombari und L. D. Stefano. „Stereo vision enabling precise border localization within a scanline optimization framework“. In: *Proceedings of the 8th Asian conference on Computer vision - Volume Part II*. Tokyo, Japan: Springer-Verlag, 2007, S. 517–527 (siehe S. 70).

[83] B. McCullagh. „Real-time disparity map computation using the cell broadband engine“. In: *Journal of Real-Time Image Processing* (2010), S. 1–7. DOI: 10.1007/s11554-010-0155-8. URL: http://dx.doi.org/10.1007/s11554-010-0155-8 (siehe S. 74).

[84] B. Meffert und O. Hochmuth. *Werkzeuge der Signalverarbeitung - Grundlagen, Anwendungsbeispiele, Übungsaufgaben*. Pearson Studium, 2004 (siehe S. 12).

[85] G. Minglun. „Near Real-Time Reliable Stereo Matching Using Programmable Graphics Hardware“. In: Hrsg. von Y. Yee-Hong. Bd. 1. 2005, S. 924–931. URL: http://doi.ieeecomputersociety.org/10.1109/CVPR.2005.246 (siehe S. 70, 121).

[86] Y. Miyajima und T. Maruyama. „A Real-Time Stereo Vision System with FPGA“. In: *Field-Programmable Logic and Applications*. Bd. 2778. Springer Berlin / Heidelberg, 2003, S. 448–457. URL: http://www.springerlink.com/content/qmtg2gulad10tdqd/ (siehe S. 71, 72).

[87] J. Mulligan und K. Daniilidis. „Predicting Disparity Windows for Real-Time Stereo“. In: *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*. London, UK: Springer-Verlag, 2000, S. 220–235 (siehe S. 28).

[88] C. Murphy, D. Lindquist, A. M. Rynning, T. Cecil, S. Leavitt und M. L. Chang. „Low-Cost Stereo Vision on an FPGA“. In: *FCCM '07: Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. Washington, DC, USA: IEEE Computer Society, 2007, S. 333–334 (siehe S. 72).

[89] G.-J. Nam, M. Yildiz, D. Z. Pan und P. H. Madden. „ISPD placement contest updates and ISPD 2007 global routing contest“. In: *Proceedings of the 2007 international symposium on Physical design*. Austin, Texas, USA: ACM, 2007, S. 167–167. DOI: http://doi.acm.org/10.1145/1231996.1232029 (siehe S. 58).

[90] NVIDIA. *NVIDIA GeForce 8800 GPU Architecture Overview*. Techn. Ber. TB-02787-001 v01. World's First Unified DirectX 10 GPU Delivering Unparalleled Performance and Image Quality. 2006 (siehe S. 50, 51).

[91] S. Park und H. Jeong. „High-speed parallel very large scale integration architecture for global stereo matching“. In: *J. Electronic Imaging* 17.1 (2008), S. 010501 (siehe S. 70).

[92] S. Park und H. Jeong. „Trellis Based Real-Time Depth Perception Chip Using Interline Constraint“. In: *New Directions in Intelligent Interactive Multimedia*.

Bd. 142. Springer Berlin / Heidelberg, 2008, S. 565–575. URL: http://www.springerlink.com/content/b5hh04685g361247/ (siehe S. 70).

[93] J. M. Perez, P. Sanchez und M. Martinez. „Real-Time Stereo Matching Using Memory-Efficient Belief Propagation for High-Definition 3D Tele-Presence Systems“. In: *CIARP '09: Proceedings of the 14th Iberoamerican Conference on Pattern Recognition*. Berlin, Heidelberg: Springer-Verlag, 2009, S. 825–833 (siehe S. 70).

[94] L. Philippe. „Robustness to Noise of Stereo Matching“. In: Hrsg. von M. John. 2003, S. 606–606. URL: http://doi.ieeecomputersociety.org/10.1109/ICIAP.2003.1234117 (siehe S. 27).

[95] F. Robotics. *Focus Robotics nDepth stereo vision*. 2010. URL: http://www.focusrobotics.com (siehe S. 75).

[96] A. Saxena, S. H. Chung und A. Y. Ng. „3-D Depth Reconstruction from a Single Still Image“. In: *Int. J. Comput. Vision* 76.1 (2008), S. 53–69 (siehe S. 4).

[97] D. Scharstein und R. Szeliski. „A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms“. In: *International Journal of Computer Vision* 47 (2001), S. 7–42 (siehe S. 24, 25, 37–39, 70, 77, 97, 117, 120, 121, 127, 128).

[98] R. Schuster und B. Braunecker. „Calibration of the LH Systems ADS40 Airborne Digital Sensor“. In: *International Archives of Photogrammetry and Remote Sensing* XXXIII (2000), S. 288–294. URL: http://gi.leica-geosystems.com/documents/pdf/ads40_calibration_whitepaper.pdf (siehe S. 88).

[99] M. Shimizu und M. Okutomi. „Sub-Pixel Estimation Error Cancellation on Area-Based Matching“. In: *International Journal of Computer Vision* 63.3 (2005), S. 207–224. DOI: 10.1007/s11263-005-6878-5. URL: http://dx.doi.org/10.1007/s11263-005-6878-5 (siehe S. 35, 41).

[100] J. Sun, H.-y. Shum und N.-n. Zheng. „Stereo matching using belief propagation“. In: *Ieee Transactions on Pattern Analysis and Machine Intelligence* 25 (2003), S. 787–800 (siehe S. 30).

[101] G. Szedo. *Two-Dimensional Rank Order Filter*. Techn. Ber. XAPP953. San-Jose, 2006, S. 17. URL: http://www.xilinx.com/support/documentation/application_notes/xapp953.pdf (siehe S. 91).

[102] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen und C. Rother. „A comparative study of energy minimization methods for Markov random fields with smoothness-based priors“. In: *Ieee Transactions on Pattern Analysis and Machine Intelligence* 30.6 (2008), S. 1068–1080. URL: %3CGo%20to%20ISI%3E://WOS:000254872500011http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4420084&tag=1 (siehe S. 30).

[103] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag Gmbh, 2010 (siehe S. 4).

[104] M. F. Tappen und W. T. Freeman. „Comparison of Graph Cuts with Belief Propagation for Stereo, Using Identical MRF Parameters“. In: *In ICCV*. 2003, S. 900–907 (siehe S. 30, 69).

[105] N. Thakoor und S. Jung. „Real-time planar surface segmentation in disparity space“. In: *In Proceedings of IEEE Computer Vision and Pattern Recognition*. 2007, S. 1–8 (siehe S. 75).

[106] C. Tomasi und R. Manduchi. „Bilateral filtering for gray and color images“. In: *Sixth International Conference on Computer Vision* (1998), S. 839–846. URL: %3CGo%20to%20ISI%3E://WOS:000075656000121 (siehe S. 21, 28, 71).

[107] F. Tombari, S. Mattoccia, L. Di Stefano und E. Addimanda. „Near real-time stereo based on effective cost aggregation“. In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. 2008, S. 1–4. URL: 10.1109/ICPR.2008.4761024 (siehe S. 70, 121).

[108] F. Tombari, S. Mattoccia und L. D. Stefano. „Segmentation-based adaptive support for accurate stereo correspondence“. In: *Proceedings of the 2nd Pacific Rim conference on Advances in image and video technology*. Santiago, Chile: Springer-Verlag, 2007, S. 427–438 (siehe S. 28).

[109] F. Tombari, S. Mattoccia, L. di Stefano und E. Addimanda. „Classification and evaluation of cost aggregation methods for stereo correspondence“. In: *CVPR*. 2008 (siehe S. 71).

[110] Valde. *Altera FPGA Co-Processors Accelerate the Performance of 3-D Stereo Image Processing*. 2010. URL: http://www.valdesystems.com/ (siehe S. 75).

[111] Valde. *webpage*. 2010. URL: http://www.valdesystems.com/ (siehe S. 75).

[112] T. Vaudrey, C. Rabe, R. Klette und J. Milburn. „Differences between stereo and motion behaviour on synthetic and real-world stereo sequences“. In: *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*. 2008, S. 1–6. URL: 10.1109/IVCNZ.2008.4762133 (siehe S. 73, 90).

[113] O. Veksler. „Fast variable window for stereo correspondence using integral images“. In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Bd. 1. 2003, pages. URL: 10.1109/CVPR.2003.1211403 (siehe S. 28).

[114] O. Veksler. „Stereo correspondence by dynamic programming on a tree“. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Bd. 2. 2005, 384–390 vol. 2. URL: 10.1109/CVPR.2005.334 (siehe S. 30).

[115] L. Wang, M. Liao, M. Gong, R. Yang und D. Nister. „High-Quality Real-Time Stereo Using Adaptive Cost Aggregation and Dynamic Programming“. In: *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. Washington, DC, USA: IEEE Computer Society, 2006, S. 798–805 (siehe S. 30, 70, 121).

[116] J. Woodfill und B. Von Herzen. „Real-time stereo vision on the PARTS reconfigurable computer“. In: *FCCM '97: Proceedings of the 5th IEEE Symposium*

on FPGA-Based Custom Computing Machines. Washington, DC, USA: IEEE Computer Society, 1997, S. 201 (siehe S. 71, 72).

[117] Q. Yang, C. Engels und A. Akbarzadeh. *Near Real-time Stereo for Weakly-Textured Scenes*. 2008 (siehe S. 69, 70, 120, 121).

[118] Q. Yang, L. Wang und R. Yang. *Real-time Global Stereo Matching Using Hierarchical Belief Propagation*. 2006 (siehe S. 69, 70, 120, 121).

[119] K. J. Yoon und I. S. Kweon. „Adaptive support-weight approach for correspondence search". In: *Ieee Transactions on Pattern Analysis and Machine Intelligence* 28.4 (2006), S. 650–656. URL: %3CGo%20to%20ISI%3E://000235253300014 (siehe S. 28).

[120] R. Zabih und J. Woodfill. „Non-parametric local transforms for computing visual correspondence". In: *Proceedings of the third European conference on Computer Vision (Vol. II)*. Stockholm, Sweden: Springer-Verlag New York, Inc., 1994, S. 151–158 (siehe S. 27, 71).

[121] C. L. Zitnick und T. Kanade. „A cooperative algorithm for stereo matching and occlusion detection". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.7 (2000), S. 675–684. URL: 10.1109/34.865184 (siehe S. 28).

Abbildungsverzeichnis

Tabellenverzeichnis

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe; ich mich nicht anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze und mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist.

Berlin, den 22.04.2013 Maximilian Buder